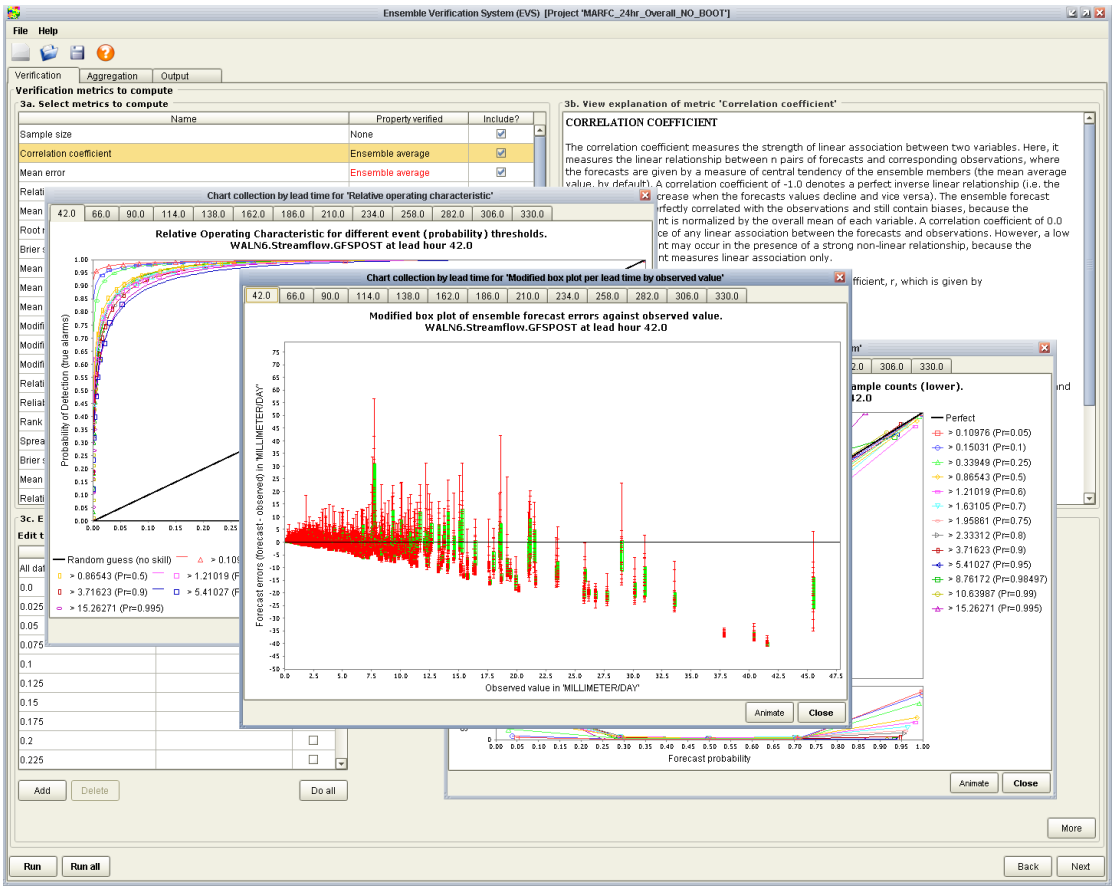
Ensemble Verification Service (EVS)

Version 5.5



Release notes

Dr. James D. Brown

Hydrologic Solutions Limited, Southampton, UK.

evs@hydrosolved.com

HSL

Preface

The Ensemble Verification Service (EVS) is a Java-based software tool originally developed by the U.S. National Weather Service's Office of Hydrologic Development (OHD) and, subsequently, by Hydrologic Solutions Limited (HSL), Southampton, UK. The software is currently developed and marketed by HSL as the Ensemble Verification System. The EVS is designed to verify ensemble forecasts of hydrologic and hydrometeorological variables, such as temperature, precipitation, streamflow and river stage. The software is intended to be flexible, modular, and open to accommodate enhancements and additions, not only by its developers, but also by its users. The EVS is "open source" software and is released under the GNU Lesser General Public License (LGPL), Version 3.0. We welcome your participation in the continuing development of the EVS toward a versatile and standardized tool for ensemble verification.

EVS Primary Point of Contact, evs@hydrosolved.com

Acknowledgments

Parts of this work were funded by NOAA's Advanced Hydrologic Prediction Service (AHPS) and by the Climate Prediction Program for the Americas (CPPA).

Disclaimer

This software and related documentation was originally developed by the National Weather Service (NWS) and, subsequently, by Hydrologic Solutions Limited (HSL), hereafter referred to as “The Developers.” Pursuant to title 17, section 105 of the United States Code this software is not subject to copyright protection and may be used, copied, modified, and distributed without fee or cost. Parties who develop software incorporating predominantly NWS developed software must include notice, as required by Title 17, Section 403 of the United States Code. The Developers provide no warranty, expressed or implied, as to the correctness of the furnished software or its suitability for any purpose. The Developers assume no responsibility, whatsoever, for its use by other parties, about its quality, reliability, or any other characteristic. The Developers may change this software to meet their own needs or discontinue its use without prior notice. The Developers cannot assist users without prior agreement and are not obligated to fix reported problems. The EVS is released under the GNU Lesser General Public License (LGPL) Version 3.0. A copy of the LGPL is provided with this distribution.

Contents

1.0 Changes from EVS 5.4 to EVS 5.5

Fix: Empty context menu displayed in the EVS file choosers [BUG]

Description

The context menu (right click) in the EVSFileChooser.java displayed an empty popup menu.

Cause

This originates from a bug in the Alloy Look and Feel (the default EVS Look and Feel) when attempting to show the context menu in the parent class of EVSFileChooser.java (javax.swing.JFileChooser.java).

Fix

Updated the EVSFileChooser.java to hide the empty context menu by setting a null menu in the superclass, JFileChooser.java, upon constructing the EVSFileChooser.java.

Notes

Tested the updated EVSFileChooser.java by attempting to activate the context menu (right click). The context menu was not shown, as expected.

Fix: Failure to propagate reference forecasts on adding or copying a VU [BUG]

Description

Upon adding or copying a Verification Unit (VU) in VerificationA.java, the new VU was not included in the list of reference forecasts available for existing VUs in VerificationB.java. The reference forecasts associated with skill metrics in VerificationB.java should include all VUs, whether from saved VUs or recently added or copied VUs.

Cause

The failure to propagate local changes in VerificationA.java to VerificationB.java. Updates were only propagated to VerificationB.java upon saving, where the updateUnitSelection method of VerificationA.java calls VerificationB.updateLocalData. However, local changes in VerificationA do not trigger VerificationB.updateLocalData, and are not, therefore, reflected in VerificationB.java. In particular, when adding or copying VUs in VerificationA.java, these VUs do not

appear in the list of potential reference forecasts for other VUs until the EVS project is saved.

Fix

Extended the VerificationB.updateLocalData method to coordinate the input VU as a reference forecast for other VUs. Added a call to VerificationB.updateLocalData in VerificationA.addDefaultUnit (to propagate the changes when adding a VU) and VerificationA.copySelectedUnit (to propagate the changes when copying a VU).

Notes

Examined the list of reference forecasts associated with skill metrics in VerificationB.java. Upon adding and copying VUs in VerificationA.java (with and without saving these changes), the local changes were correctly propagated to VerificationB.java.

Fix: Setting advanced value conditions before aggregation parameters [BUG]

Description

Upon attempting to save both advanced aggregation parameters and advanced value conditions in the MoreVerificationWindowDialog.java, an exception was thrown, indicating that the aggregation parameters could not be saved. As the value conditions apply to aggregated quantities, the aggregation parameters must be set before the value conditions.

Cause

The saveData method in MoreVerificationWindowDialog.java attempted to save the value conditions before the aggregation parameters. Consequently, the setResolution method in VerificationUnit.java threw an exception, indicating that the verification resolution could not be set until the value conditions were removed.

Fix

Updated the saveData method in MoreVerificationWindowDialog.java to set the aggregation parameters before any date or value conditions.

Notes

Tested the MoreVerificationWindowDialog.java by adding aggregation conditions and value conditions in different combinations. No exceptions were thrown when saving these parameters.

Fix: Failure to preserve local data on metrics when copying VUs [BUG]

Description

Upon copying a Verification Unit (VU) in the first window of the Verification stage, the local parameters stored in VerificationA.java and VerificationB.java should be copied from the source VU to the new VU. In practice, only those local parameters stored in VerificationA.java were copied to the new VU. Thus, any local parameters stored in VerificationB.java were not preserved in the copied VU (unless they matched the saved parameters). The desired behavior is to preserve all local data entered in the GUI upon saving or copying a VU.

Cause

The failure to associate any local parameters from VerificationB.java with the copied VU. Specifically, the copySelectedUnit method of VerificationA.java failed to request a deep copy of the local parameters associated with the source VU in VerificationB.java and store them with the copied VU.

Fix

Updated VerificationB.java to add a new method, copyLocalData, which deep copies the local data associated with a source VU and assigns it to a copied VU. This required several additional methods to support the deep copying of local data in VerificationB.java, notably two new methods in EVSUtilities.java for copying primitive arrays (isPrimitiveArray and copyPrimitiveArray) and a deepCopy method in DisplayScoreDecompPar.java. Updated copySelectedUnit in VerificationA.java to call VerificationB.copyLocalData, which copies the verification metrics in the local store for the source VU and associates them with the copied VU.

Notes

Tested the mechanism to copy local parameters in VerificationB.java and associate them with a copied VU, including for a source VU *with* saved metrics and for a source VU *without* saved metrics. In both cases, the local parameters in VerificationB.java were preserved in the copied VU.

Fix: Errors displayed from modal dialogs may disappear behind those dialogs [BUG]

Description

Error messages generated from within modal dialogs and displayed by the ExceptionHandler.java were occasionally blocked by the modal dialog and became

hidden, freezing the GUI until the error message was unblocked (e.g. by cycling through open windows).

Cause

The failure to associate an error message with a parent dialog when calling the displayException method of ExceptionHandler.java. When the parent dialog is modal, the error message may be blocked by the modal dialog.

Fix

Updated all calls to the displayException method in ExceptionHandler.java to set the parent dialog for error messages, including those generated from within a modal dialog, such as MoreInputDialog.java

Notes

Tested the display of errors generated from within modal dialogs and displayed through the ExceptionHandler.java. The error messages were correctly associated with the parent dialog and no blocking was observed.

Fix: Failure to reset unconditional list of items in SearcheableComboBox [BUG]

Description

Once a single item has been identified and selected using the SearcheableComboBox.java, the original (unconditional) list of items should appear on starting a new search. In practice, the last conditional (searched) list appeared.

Cause

The failure to update the original (unconditional) list in SearcheableComboBox.java once a single item has been identified and selected. Specifically, the setSelectedItem method should delete the conditional list and set the active list of items to the unconditional list.

Fix

Overrode the setSelectedItem method of the superclass, JComboBox.java, and deleted the last conditional list, replacing the active list with the unconditional list of items.

Notes

Conducted various searches with the SearcheableComboBox.java, selecting one item from a conditional list and then starting a new search. Upon starting a new

search, the active list was returned to the unconditional list, as required. All other expected behaviors were retained.

Fix: Failure to confirm partial entries in advanced options table upon saving [BUG]

Description

When entering parameter values in tables throughout the EVS, the default behavior is to accept partial entries upon saving (i.e. entries for which the cell editor remains open). For the table of paired options under the advanced parameter dialog, MoreInputDialog.java, the table cell editor remained open upon closing the dialog. Consequently, any partial entries were not confirmed and saved.

Cause

Upon closing the MoreInputDialog.java, the table cell editor for the advanced table of pairing options was not properly closed by confirming all partial entries.

Fix

Updated the saveData method of MoreInputDialog.java to close the table cell editor for the table of advanced options by calling javax.swing.table.TableCellEditor.stopCellEditing.

Notes

Tested the MoreInputDialog.java by leaving several entries open in the table of advanced pairing options. Upon closing the dialog, the partial entries were accepted and the table cell editor closed, as expected.

Fix: Failure to recompute climatological thresholds in an Aggregation Unit [BUG]

Description

Upon changing one or more Verification Units (VUs) within an Aggregation Unit (AU) and saving the EVS project file, any further attempt to re-run the AU would fail if the component VUs comprised climatological probability thresholds.

Cause

When executing a VU, the real values associated with climatological probability thresholds are computed from the verifying observations and stored temporarily within those thresholds (DoubleProcedureParameter.java). Upon saving the EVS project file, these thresholds (and hence the real values) are overwritten. This was not properly factored into the spatial aggregation of VUs within an AU. Specifically,

the computeMetricsByAveraging method of the AggregationUnit.java failed to recompute VUs for which existing results were available. This generated a run-time exception from the aggregate method of the DoubleProcedureParameter.java, indicating that real-valued thresholds were missing for one or more of the component VUs.

Fix

Updated the computeMetricsByAveraging method of AggregationUnit.java to consistently recompute VUs, including those VUs for which verification results are available.

Notes

For an EVS project file containing an AU, reproduced the exception conditions by: 1) executing the AU; 2) modifying one of the component VUs in the AU and saving the EVS project; and 3) re-running the AU. The AU ran successfully on both occasions.

Fix: Loss of saved data when swapping identifiers between Verification Units [BUG]

Description

When renaming multiple Verification Units (VUs) simultaneously, all of the verification metrics and parameter values associated with the renamed VUs were lost. Specifically, they were lost for those VUs whose identifiers were swapped.

Cause

When two or more VUs were renamed simultaneously, the synchonizeVUNames method of Verification_B.java failed to re-assign the locally saved information in Verification_B.java (on verification metrics and associated parameter values) to each of the renamed VUs.

Fix

Updated the synchonizeVUNames method in Verification_B.java to correctly re-assign all locally saved parameters to the renamed VUs.

Notes

Tested the fix by renaming multiple VUs simultaneously; checked the assigned verification metrics and parameter values for the renamed VUs, which were all correct.

Fix: Allow for reading of tarred/zipped archives of ASCII forecast files [ENHANCE]

Description

In order to preserve disk space, forecast files in PI-XML and EVS ASCII formats may be tarred and compressed. Allowing the EVS to read these compressed archives directly would minimize storage space and avoid any additional administration of the files for verification purposes (i.e. untarring and unzipping temporarily).

Cause

The inability to read forecast files in an ASCII format (specifically PI-XML and EVS ASCII files) from a compressed archive.

Fix

Extended PairedDataSource.java to support the reading of forecast archives in a tarred and gzipped format. Separated the reading of forecast files into two methods, processForecastFile and processForecastArchive, where the latter processes tarred and gzipped archives. Extended ASCIIFileIO.java and PublishedInterfaceXMLIO.java to read forecasts directly from a java.io.BufferedReader.java. Implemented BufferedArchiveReader.java, which extends java.io.BufferedReader, and overrides the close method to remain open across a sequence of entries in a forecast archive. The BufferedArchiveReader.java is closed upon request, once all entries in a forecast archive have been processed or an exception has been thrown. Updated the computeAndSetPairs method of PairedDataSource.java to distinguish between a single forecast file and a forecast archive, calling the appropriate reader in each case.

Notes

Tested file reading of individual forecast files in PI-XML and ASCII formats and tarred and gzipped archives of PI-XML and ASCII files. The forecasts were processed correctly in all cases.

Fix: Added an ensemble quantile-quantile diagram [ENHANCE]

Description

Added a quantile-quantile diagram to the EVS. The new metric forms an average of the order statistics of the individual ensemble members and compares them to the corresponding order statistics of the observations. The metric is currently only accessible via the EVS project file and has not been added to the GUI.

Cause

Lack of support for evaluating climatological biases in ensemble forecasts with a quantile-quantile diagram.

Fix

Created a new class that generates the metric results, EnsembleQQDiagram.java, and a default class to plot the results (QQPlot.java). Updated the Chartfactory.java to handle the results from the EnsembleQQDiagram.java using the QQPlot.java. The new metric is configured through the EVS project file and is not currently available in the GUI. The metric is configured within the <metrics> block of the EVS project file:

```
<metric>
        <name>EnsembleQQDiagram</name>
        <forecast_type_parameter>regular</forecast_type_parameter>
        <unconditional_parameter>false</unconditional_parameter>
        <bootstrap_parameters>
                <technique>None</technique>
        </bootstrap_parameters>
</metric>
```

Notes

Configured the EnsembleQQDiagram.java for a test case and compared the results to a benchmark.

Fix: Increment the Location identifier when adding or copying a VU [ENHANCE]

Description

Upon copying a Verification Unit (VU) in VerificationA.java, the new VU was named by incrementing the Environmental variable identifier of the old VU. When a new VU is added, the updateLocalData method in AggregationA.java refreshes the list of VUs available for aggregation. An Aggregation Unit (AU) cannot be formed from VUs with different Environmental variable identifiers. Thus, upon copying an existing VU, the new VU failed to display in AggregationA.java until the Environmental variable identifier was updated *and* the changes saved. Although expected behavior, incrementing the Location identifier is preferred, as this would immediately propagate the copied VU to AggregationA.java, without the need to update and save.

Cause

The copySelectedUnit method of VerificationA.java creates a new VU by incrementing the Environmental variable identifier of the old VU. Until the

Environmental variable identifier is updated, and the changes saved, the new VU cannot appear in the list of VUs available for aggregation in AggregationA.java.

Fix

Updated the copySelectedUnit method in VerificationA.java to increment the Location identifier upon copying a VU. For consistency, also updated the addDefaultUnit method of VerificationA.java to increment the Location identifier when adding a new VU (although an empty VU cannot appear in AggregationA.java).

Notes

Upon adding and copying a VU, the Location identifier is now updated. A copied VU immediately appears in the list of VUs available for aggregation in AggregationA.java.

Fix: Allow for filtering of forecast files within an archive [ENHANCE]

Description

When reading forecast files from an archive, such as a tarred and compressed archive, all files are processed and interpreted as valid forecast files. In practice, the archive may contain files that should not be interpreted as forecast files. Filtering input files against a prescribed pattern (e.g. .xml) is already supported when reading from a directory, but not from an archive.

Cause

The inability to read selected files within an archive by filtering the names of the files against a prescribed pattern.

Fix

Implemented a new class, FileArchiveDataSource.java, to identify a file source that comprises an archive. The FileArchiveDataSource.java extends the FileDataSource.java and allows for the identification of a file filter to be applied within the archive. Updated the PairedDataSource.java to support the reading of forecast files from a FileArchiveDataSource.java. Also updated the MoreInputDialog.java to allow for the identification of a filter string within the GUI (using the Other options tab of the Additional options dialog) and updated the ProjectFileIO.java to allow for reading and writing of the filter string to an EVS project file.

Notes

Tested the new filtering option by reading files from a forecast archive and prescribing various filter patterns. The archive was filtered correctly and the filter was

edited and displayed successfully within the GUI, stored correctly in the EVS project file, and retrieved correctly upon re-opening the EVS project file.

2.0 Changes from EVS 5.3 to EVS 5.4

Fix: Incorrect updating of local data in the EVS GUI [BUG]

Description

Local data is saved in each window of the EVS GUI upon entry for the active VU. When selecting a metric in the second window of the Verification stage and updating the local data for that metric, the updates were incorrectly applied to the same metric across all VUs in the active project, rather than the active VU only.

Cause

The saveLocalData method of GUIInterface.java is implemented by VerificationB.java and other windows in the EVS GUI. Rather than obtaining the active VU from an input variable, the saveLocalData method must call the getSelectedUnit method of VerificationA.java. However, depending on the context in which saveLocalData is called, updates may be required to a VU other than the active VU returned by the getSelectedUnit method. For example, a ListSelectionListener is registered with the table of VUs in VerificationA.java. Upon selecting a new VU, the updateUnitSelection method of VerificationA.java calls the saveLocalData method of VerificationB.java. This call is made only when the ChangeEvent indicates that the value is adjusting. Under these conditions, the getSelectedUnit method returns the currently selected VU, and not the previously selected VU for which the local data must be saved. Rather than obtaining the appropriate VU indirectly, the saveLocalData method of GUIInterface.java should obtain this VU as an input variable. Similarly, the showLocalData method of GUIInterface.java should obtain the appropriate VU as an input variable.

Fix

Updated the saveLocalData and showLocalData methods of the GUIInterface.java and all implementing classes to include the VU for which any changes are required as an input variable, avoiding the need to obtain this VU indirectly.

Notes

Upon selecting a metric in the second window of the Verification stage and updating the local data for that metric, the updates are no longer applied to the corresponding metric associated with other VUs in the active project.

Fix: Failure to conduct spatial aggregation for probability thresholds [BUG]

Description

When forming an AU by averaging the verification results across several VUs, the verification results for a ThresholdMetric should be averaged together with the verification thresholds. When computing the VUs and AUs together, both were computed correctly. However, when separating these stages (i.e. computing the VUs first and then computing the AU), an exception was thrown for each ThresholdMetric, indicating that the verification thresholds could not be aggregated.

Cause

The aggregate method of DoubleProcedureParameter.java computes an aggregate verification threshold from a vector of input thresholds (instances of DoubleProcedureParameter) using a specified aggregation function and corresponding vector of weights (one per threshold). Alongside the threshold value, each input threshold comprises two logical parameters, one indicating whether the thresholds are probabilities and another indicating whether the thresholds are “main” thresholds (and should be included in the plotting). These parameters should be maintained in the aggregated output. In practice, however, they were not maintained. This resulted in a conflict in the aggregated DoubleProcedureParameter and, ultimately, an exception upon attempting to aggregate the input thresholds.

Fix

Updated the aggregate method of DoubleProcedureParameter.java to ensure that the aggregated verification thresholds comprise the same logical parameters as the input thresholds.

Notes

Tested the updates by conducting aggregation both prior to and after forming the VUs. In both cases, the aggregation was conducted without exceptions.

Fix: Failure to delete verification pairs when changing advanced parameters [BUG]

Description

Upon executing a VU for the first time, the verification pairs are computed and stored for future runs. In some cases, a parameter is changed that implies the verification pairs should be deleted and recomputed. However, several advanced parameter

options that should trigger existing pairs to be deleted and recomputed, failed to trigger this update.

Cause

Upon changing a parameter that implies the existing verification pairs should be deleted and recomputed, the appropriate setter method in VerificationUnit.java determines whether the input parameter differs from the existing parameter. If a difference is identified, the old pairs are deleted. However, several of the setter methods for advanced parameters failed to check for equality *before* assigning the parameters, instead making the check afterwards (when the parameters are necessarily equal). The check for equality was conducted incorrectly by multiple setter methods in VerificationUnit.java, namely: setFcstAggStartTimeUTC, setObsAggStartTimeUTC, and setFcstAggStartLeadHour. These setter methods correspond to the advanced parameters in 2b. Identify input data sources > More > Pairing options, namely: “Set time for aggregation of observations [hours, UTC]”, “Set time for aggregation of forecasts [hours, UTC]”, and “First lead time for aggregation of forecasts [hours]”, respectively. In addition, the checks for equality were conducted incorrectly by setPairedAggStartTimeUTC, setPairedAggStartLeadHour. These methods correspond to the advanced parameters in 2c. Set time parameters > More > Aggregation, namely: “Aggregation start hour UTC [0, 23]”, and “Aggregation start lead hour”, respectively. When editing any of these parameters for an EVS project file with existing verification pairs, these changes failed to delete the existing pairs.

Fix

Updated the following setter methods of VerificationUnit.java to ensure that the input parameter is checked for equality with the existing parameter before assigning the input parameter to the existing parameter: setPairedAggStartTimeUTC, setFcstAggStartTimeUTC, setObsAggStartTimeUTC, setPairedAggStartLeadHour, and setFcstAggStartLeadHour.

Notes

Tested each of the updated methods by assigning parameter values that differed from the existing values. In each case, an update was triggered and the existing pairs deleted, as expected.

Fix: Inefficient saving of local data in the EVS GUI [BUG]

Description

The second window of the Verification stage saves local data upon entry in the EVS GUI. The local data is only saved to an EVS project file upon an explicit request to save the EVS project. A listener is registered to the threshold table for each verification metric, which saves any edits to the thresholds in the local data store. Upon editing a given cell in the table, the listener was triggered for *every* cell in the table, rather than once for the edited cell.

Cause

The showLocalData method of VerificationB.java registered a CellEditorListener for each cell in the threshold table for a given verification metric. Upon editing a table cell, the listener fired a ChangeEvent for every cell in the table, which called the saveLocalData method of VerificationB.java. Rather than associating a listener with each cell, a listener should be associated with each column in the threshold table.

Fix

Updated the showLocalData method of VerificationB.java to avoid registering a CellEditorListener with each cell in the threshold table. Updated the setTables method of VerificationB.java to register a CellEditorListener for each column class in the threshold table, which calls the saveLocalData method once for each edit.

Notes

Added a print statement to the saveLocalData method of VerificationB.java to identify the number of calls to this method before and after the updates. After the updates, a single edit resulted in a single call to saveLocalData, which is the expected behavior.

Fix: New options to sub-sample verification pairs for sensitivity testing [ENHANCE]

Description

Verification results are sensitive to the paired forecasts and observations available. Forecasts and observations from consecutive lead times and from adjacent locations are typically related to each other or “statistically dependent”. If the verification pairs contain shared information, the effective sample size is smaller than the nominal sample size. Particularly when the nominal sample size is small (e.g. extreme events), the verification results may be over-sensitive to a small number of observed events that are shared across multiple forecast issue times. In order to explore these

sensitivities, the EVS was enhanced to allow for sub-sampling of verification pairs at prescribed intervals; that is, thinning of data to include only those forecast issue times that meet prescribed constraints.

Cause

The inability to subsample pairs at a prescribed start index and frequency, in order to explore the sensitivity of the verification results to the sample data available and, specifically, to shared information across multiple dates.

Fix

Updated PairedData.java to include a new method, getThinnedPairs, which takes a paired dataset and returns a thinned dataset, comprising a sub-sample of the original pairs. The pairs are sub-sampled by forecast issue time with a prescribed, zero-based, start index and an interval between forecasts (i.e. sample every nth forecast). Updated VerificationUnit.java to set, store, and return the start index and frequency, and to thin the paired data associated with a Verification Unit upon request. The thinning is applied to the conditional pairs (i.e. the subset of pairs remaining after all other conditions on dates and variable values have been applied). Finally, updated ProjectFileIO.java to read and write the thinning parameters, including the start index and frequency. These parameters are included in a <thinning> tag within the <verification_window> of the EVS project file. For example:

```
<thinning>
      <start_index>0</start_index>
      <frequency>3</frequency>
</thinning>
```

Notes

Tested the new functionality by specifying various combinations of start index and frequency and validating the resulting selection of conditional pairs. The pairs were subsampled correctly and the thinning parameter were written to the EVS project file and read correctly.

3.0 Changes from EVS 5.2 to EVS 5.3

Fix: Incorrect labeling of the range axis in the plot of relative mean error [BUG]

Description

The range axis in the default plot of the relative mean error (RME) was expressed in real units (i.e. where real units are defined in the EVS project file). However, the RME measures the fractional bias of the ensemble mean forecast and is, therefore, dimensionless.

Cause

The RMEPlot.java incorrectly implemented RealValuedPlot.java, which requires all implementing classes to add the real units (if available) to the range axis.

Fix

Removed the incorrect inheritance of RealValuedPlot.java from RMEPlot.java and removed the corresponding implemented method, setRealUnits.

Notes

Ran the benchmark tests and verified that the plots of RME no longer label the range axis with real-valued units.

Fix: Failure to set forecast file filter on loading an EVS project file [BUG]

Description

The forecast file filter is an advanced option under 2b. Identify input data sources > More > Other options, which allows a directory to be searched for only those forecast files that match the specified filter text (e.g. .xml). When loading a new EVS project file, the filter text was not always synchronized with the corresponding local parameter in the GUI. This led to a conflicted state whereby the stored parameter was removed, which further triggered the verification pairs to be removed (which are invalidated by a change in this parameter). It also led to a run failure or an incorrect run, as the forecast directory was no longer screened with the specified filter.

Cause

When loading a new EVS project file for which a filter was defined, the parameter was not immediately synchronized with the local parameter in the GUI. Rather, it was only synchronized when opening the Other options dialog. Thus, unless the Other options dialog was opened between loading a project file and executing a run, a

conflicted state arose, whereby the local parameter remained null. Since an automatic save operation is triggered before each run (which sets the saved parameters with the local parameters), the resulting (non-null) forecast file filter was removed prior to the run. This resulted in a failed or invalid run.

Fix

Upon loading an EVS project file, the updateLocalData method of VerificationA.java is used to synchronize the local store of parameters in the GUI with those in the EVS project file. This method now calls the corresponding updateLocalData method of the MoreInputDialog.java, which stores the local parameter for the forecast file filter.

Notes

Repeated the failure scenario whereby a forecast file filter was previously removed upon loading an EVS project file. The filter is no longer removed.

Fix: Failure to remove non-default pairing options once set in GUI [BUG]

Description

The pairing options window of the additional options dialog (MoreInputDialog.java) contains several options for pairing forecasts and observations. The dialog is accessible via the “More” button from 2b. Identify input data sources in the first window of the GUI. Several of these options do not have default values (i.e. they are empty). Upon setting a non-default value, it was subsequently impossible to remove that value by setting an empty string (i.e. the empty string was not accepted), although other non-default values (i.e. non-empty strings) were accepted.

Cause

Upon finding an empty string for any of the pairing options that allow empty values, the appropriate methods in VerificationUnit.java were not called to remove the existing values of those parameters.

Fix

Upon encountering an empty string in MoreInputDialog.java for any parameters that allow an empty string, the appropriate methods are now called in VerificationUnit.java to remove the existing parameter value. For convenience, additional methods were defined in VerificationUnit.java to delete an existing parameter value and set the default (empty) value.

Notes

Repeated the scenario whereby empty strings were not allowed once a non-empty string had been set. Empty strings are now accepted and the parameters returned to their default (empty) status upon request.

Fix: Automatic threshold entries retained in GUI when closing a project [BUG]

Description

When closing an EVS project (i.e. closing a project, creating a new project or opening a new project), the entries for automatic threshold generation in the second window of the Verification stage were retained in the GUI. The desired behavior is to clear all parameter entries, including the entries for automatic threshold generation, when closing a project.

Cause

The clearDisplay method of VerificationB.java failed to clear the entries for automatic threshold generation when requested.

Fix

Updated the clearDisplay method of VerificationB.java to clear the entries for automatic threshold generation when requested.

Notes

Repeated the scenario whereby automatic thresholds entries were retained on closing a project. The entries are no longer retained.

Fix: Default parameter inputs remain visible when no metric is selected [BUG]

Description

When selecting a verification metric in the second window of the Verification stage and opening a new project, the metric selection is returned to default, i.e. no metric selected (expected behavior). However, the default parameter inputs for the previously selected metric remain visible and disabled. Since the default parameter inputs vary with metric, the expected behavior is to show no inputs when no metric is selected.

Cause

In the absence of a metric being selected, the setParBoxEnabledState method of VerificationB.java failed to hide the default parameter inputs.

Fix

Updated the setParBoxEnabledState method of VerificationB.java to hide the default parameter inputs when no metric is selected.

Notes

Repeated the scenario whereby the default parameter inputs were disabled but remained visible when no metric was selected. The default parameter inputs are now hidden, as expected.

Fix: Local parameter values retained on closing/opening/creating a project [BUG]

Description

When loading a new project file, not all of the local data from a previously opened project were cleared from the second window of the Verification stage. For example, after manually removing a metric from a loaded EVS project file and re-loading the project, the metric was retained in the GUI, when the proper behavior should be to clear all local data and display the newly opened project (and thus not display the eliminated metric).

Cause

A failure to consistently clear all local data (from all windows) when calling the newProject, closeProject and openProject methods of EVSMainWindow.java.

Fix

Created a new method, clearAllLocalData, in EVSMainWindow.java to clear the local data from all windows on request. Updated the newProject, closeProject and openProject methods of EVSMainWindow.java to call clearAllLocalData.

Notes

Repeated the scenario whereby local data were retained upon closing a project, creating a new project or opening an existing project. The local data are no longer retained.

Fix: Inconsistent application of date/value conditions to climatology [BUG]

Description

When determining real-valued thresholds from climatological probabilities, there are two options for selecting climatological observations, namely the paired observations and the original (unpaired) observations. In the EVS GUI (2b. Identify input data

sources > More > Other options), any date and value conditions may be applied separately to the climatological observations (versus the verification pairs). However, when electing to use the paired observations to determine climate thresholds, any separate control on the use of date or value conditions was ignored, i.e. any date or value conditions were applied automatically to the climatological observations unless the option to use the original (unpaired) observations was also selected.

Cause

A failure to separate the application of date and value conditions from the source of data used to determine the climate thresholds. Specifically, the getConditionedPairs method of VerificationUnit.java ignored any request to apply or not apply date and value conditions to the climatological observations unless those observations comprised the original (unpaired) observations.

Fix

Updated the getConditionedPairs method of VerificationUnit.java to apply or not apply date and value conditions independently of the data source to which those conditions apply.

Notes

Repeated the scenario whereby the choice to ignore date or value conditions in determining climate thresholds was only applied when using the original (unpaired) observations. It is now applied independently of the data source used to determine the climate thresholds.

Fix: Plot range incorrectly fixed across components of score decompositions [BUG]

Description

Graphical outputs from the EVS are grouped for plots that span multiple forecast lead times and for plots that comprise score decompositions. In both cases, the range axes of the grouped plots are fixed according to the overall minimum and maximum values. However, while useful for viewing common plots across multiple forecast lead times, this was not intended for score decompositions, where the score components have different bounds and interpretations.

Cause

The failure to distinguish between groups of plots that comprise multiple forecast lead times, whose range axes should be fixed across plots, versus groups of plots that comprise score decompositions, whose range axes should not be fixed across plots.

Specifically, the generateProducts method of the ProductGenaratorDialog.java failed to make this distinction, and fixed the range axes for both types of plot groupings.

Fix

Updated the generateProducts method of the ProductGenaratorDialog.java to fix the range axes for plot groupings that comprise multiple forecast lead times and not for groupings that comprise score decompositions.

Notes

Generated graphical outputs for both plot groupings (i.e. multiple forecast lead times and score decompositions), and the range axes are no longer fixed for groupings that comprise score decompositions.

Fix: Failure to re-load project file with null forecast data source [BUG]

Description

Upon saving an EVS project file with a null data source, the project file failed to re-load.

Cause

The readVerificationUnit method of ProjectFileIO.java expected a non-null forecast data source. Upon encountering a null data source, a NullPointerException was thrown. This resulted in the failure to read an EVS project file with an empty forecast data source, yet such a file could be created within the EVS.

Fix

Updated the readVerificationUnit method of ProjectFileIO.java to support the reading of EVS project files with an empty forecast data source.

Notes

Attempted to read an EVS project file with an empty forecast data source. The file was read correctly.

Fix: Failure to correctly apply combinations of date and value conditions [BUG]

Description

When defining conditions on variable value for a Verification Unit (VU) other than the current VU, the value conditions were not applied correctly when date conditions were also defined.

Cause

The getConditionedPairs method of VerificationUnit.java failed to correctly apply combinations of date and value conditions when using value conditions for variables other than the current VU. For example, when defining a combination of date and value conditions for temperature, the value conditions were not applied correctly when referencing secondary variables, such as precipitation. Specifically, the getConditionedPairs method failed to apply the value conditions to the unconditional pairs for the secondary variable (e.g. precipitation), instead applying these to the conditional pairs. This was contrary to the design of getConditionedPairs, namely that all conditions are collated and merged before being applied to the unconditional pairs, i.e. the pairs are conditioned once rather than sequentially for each condition.

Fix

Updated the getConditionedPairs method of VerificationUnit.java to correctly apply combinations of date and value conditions to the unconditional pairs.

Notes

Checked the conditional pairs following the application of combined date and value conditions, including value conditions on secondary variables (e.g. precipitation when verifying temperature), and the conditional pairs were constructed correctly.

Fix: Failure to consistently set/clear the multiplier for target attribute units [BUG]

Description

Converting between measurement units is achieved by specifying the existing units, the target units and, for those unit conversions not recognized by the EVS, a multiplier to arrive at the target units from the existing units (“Multiplier for target units”). These inputs are available in the Additional options dialog of 2b. Identify input data sources. For unit conversions that are recognized by the EVS (i.e. conversions that appear in MeasurementUnitsChangeLibrary.java), the multiplier to arrive at the target units is displayed in the Additional options dialog upon selecting the unit conversion. However, the multiplier was not displayed or cleared consistently. For example, it was only displayed in the Additional options dialog upon re-opening the dialog, having previously defined the unit conversion. Also, an existing multiplier, displayed in the Additional options dialog (“Multiplier for target units”), was not automatically cleared when modifying an earlier unit conversion.

Cause

Inadequate logic for setting and clearing the “Multiplier for target units” in the Additional options dialog of 2b. Identify input data sources. Specifically, the setValueAt method of the ScaleTableModel.java failed to update the multiplier upon selecting a unit conversion from the MeasurementUnitsChangeLibrary.java or clear the multiplier when updating either the existing attribute units or the target units.

Fix

Updated the setValueAt method of ScaleTableModel.java to properly set and clear the multiplier for target units when identifying or editing unit conversions.

Notes

Added, edited, and removed unit conversions between existing and target attribute units. The multiplier for target units was updated or cleared, as appropriate.

Fix: Incorrect naming of conditional paired file [BUG]

Description

The conditional pairs were written to a file whose name was derived from the file name assigned to the unconditional pairs, rather than the unique identifiers of the VU to which those pairs applied. Consequently, when using a common source of unconditional pairs across several VUs or several projects, the file name assigned to the conditional pairs was not unique, and existing conditional pairs were overwritten.

Cause

When writing conditional pairs, the file name was derived from the name given to the unconditional pairs. Thus, when manually editing the paired data source in an EVS project file (e.g. to use common pairs across several projects), the conditional pairs were written to a file whose name was a derivative of that file and not the identifiers associated with the VU to which the conditional pairs applied. For the same reason, when using a common source of unconditional pairs across several VUs or several projects, the file name for the conditional pairs was no longer unique and could overwrite existing conditional pairs. The file name was assigned by the writePairs method of VerificationUnit.java, to which a fix was applied.

Fix

Added a new method getNameForPairedFile to VerificationUnit.java to return an appropriate name for the paired file, based on the unique identifiers of the VU from

which the method is called. Updated the computeAndSetPairs method of PairedDataSource.java, as well as the writePairs method of VerificationUnit.java, to call getNameForPairedFile. Previously, the writePairs method used the file name of the unconditional paired data source as the basis for the conditional pairs, rather than using the (unique) combination of identifiers available for a given VU.

Notes

Manually edited the file data source for the unconditional pairs in an existing project file. Upon writing the conditional pairs, the file name was derived from the unique identifiers of the VU to which the unconditional pairs applied, and not the file name of the unconditional pairs.

Fix: Incorrect handling of non-unique identifiers for new Verification Units [BUG]

Description

Each VU has a unique combination of identifiers. Upon changing the name of a default VU to a combination of identifiers that already exists, an appropriate exception is thrown. However, when changing several VUs simultaneously and creating two or more non-unique identifiers among the *new* names, an unhandled exception is thrown.

Cause

The saveData method of VerificationA.java compares the identifiers of existing VUs with the proposed identifiers for new VUs. If any combination of identifiers is non-unique, an exception is thrown. However, there were no checks to ensure that the proposed identifiers were unique among the list of all proposed identifiers. Thus, if several VUs were altered simultaneously, non-unique identifiers were possible. This resulted in an unhandled exception (a NoSuchElementException), rather than a handled exception and warning message.

Fix

Updated the saveDate method of VerificationA.java to ensure that all proposed identifiers are unique, as well as ensuring that the proposed identifiers do not match any existing identifiers that will remain following the updates.

Notes

Tested the exception handling following the updates. When changing several VUs simultaneously and creating non-unique identifiers among the updated names, an appropriate exception is now thrown.

Fix: Inconsistent return value from hasDateCondition in VerificationUnit.java [BUG]

Description

A date condition comprises a collection of individual conditions applied to the dates associated with the verification pairs. When a date condition is defined for a given VU, it may be modified to clear some or all of the individual conditions. When clearing all conditions, an empty date condition remains. The hasDateCondition method of VerificationUnit.java should only return true when a date condition exists and is not empty. Previously, the method returned true when an empty date condition was defined.

Cause

Inconsistent behavior of the hasDateCondition method of VerificationUnit.java when an empty date condition is available for a given VU. Specifically, the method returned true if an empty date condition was defined but returned false if no date condition was defined. The desired behavior is to return true only if a date condition exists and it is *not* empty.

Fix

Updated the hasDateCondition method of VerificationUnit.java to return true only when a date condition has been defined and the date condition is not empty.

Notes

Tested the updated method. The method now returns true only when a date condition has been defined and is not empty.

Fix: Failure to save local data in all windows upon changing the active VU [BUG]

Description

User inputs to the EVS GUI are initially saved as local data. Local data are retained for display and are not validated or saved permanently. Following a request to save an EVS project, the local data are validated and saved to a project file. User inputs are required across multiple windows. When changing the selection of VUs in the first window of the Verification stage, the local data should be saved across all windows and re-displayed upon request (i.e. upon re-selecting a VU). However, the local data was not consistently saved across all windows. Specifically, the selection of metrics and their associated parameter values, as well as the configuration of Aggregation

Units (AUs), were not retained upon changing the active VU. This required a user to explicitly save the current project before changing the active VU.

Cause

The failure to consistently store local data upon selecting a different VU in the first window of the Verification stage. Specifically, the updateUnitSelection method of VerificationA.java was incorrectly calling the updateLocalData method of VerificationB.java and AggregationA.java (which updates the locally stored parameter values with previously saved values), rather than calling saveLocalData.

Fix

Changed the updateUnitSelection method of VerificationA.java to call the saveLocalData method of VerificationB.java and AggregationA.java before calling the showLocalData method of those classes.

Notes

Tested the updated method. The local data was stored and re-displayed consistently across all windows when changing the active VU.

Fix: Failure to consistently use all historical observations for climate thresholds [BUG]

Description

Climatological probabilities may be determined from all historical observations or only from those observations associated with the verification pairs. Additionally, the observations may be constrained by date and value conditions. In the GUI, the use of unconditional observations is configured via 2b. Identify input data sources > More > Other options > Use all observations (not just paired) to determine climate thresholds. This option is applied correctly when computing verification metrics from an existing paired dataset, but was not applied correctly when computing verification metrics for the first time. Here, the unconditional observations were (incorrectly) constrained by the start and end dates of the verification window.

Cause

The computeAndSetPairs method of PairedDataSource.java used unconditional observations that were constrained by the start and end dates of the verification window, rather than the full set of unconditional observations. In contrast, when re-reading verification pairs, the getPairs method of VerificationUnit.java used the full set of unconditional observations.

Fix

Updated the computeAndSetPairs method of PairedDataSource.java to associate the full set of unconditional observations with the paired data upon initial construction. In future, the unconditional observations may be stored directly in the paired data file.

Notes

Tested the updated method. The unconditional observations are now correctly associated with the verification pairs upon construction.

Fix: Failure to synchronize an AU upon removing the additional ID from a VU [BUG]

Description

When changing the identifiers of a VU, these identifiers should be updated for all VUs and AUs that depend on the renamed VU. For example, AUs that comprise a renamed VU should be updated to use the renamed VU. However, when attempting to remove the additional ID for a VU contained within an AU, an unhandled exception was thrown and the AU was left in an inconsistent state.

Cause

The synchronizeVUNames method of AggregationA.java attempts to synchronize any changes in VU identifiers for AUs that comprise the renamed VU. This method correctly handled changes in the location ID (a required identifier) and in the additional ID (an optional identifier), but failed to handle the removal of the additional ID, for which no method was defined in VerificationUnit.java.

Fix

Updated VerificationUnit.java to include a new method, clearAdditionalID, which removes the additional ID associated with a VU. Updated the synchronizeVUNames method of AggregationA.java to handle the removal of an additional ID by calling the clearAdditionalID method of VerificationUnit.java.

Notes

Tested the removal of an additional ID for a VU that was contained within an AU. The AU was updated correctly to include the renamed VU (without the additional ID).

Fix: Incorrect metadata files generated for Aggregation Units (AUs) [BUG]

Description

The numerical outputs from the EVS comprise graphical outputs and numerical outputs in XML format. Separate XML files are written for the verification results and for metadata, which includes the verification thresholds and sample counts. When writing the metadata for aggregation units, the thresholds and sample counts were not correct. Specifically, they referred to the first VU in the spatial aggregation and not the AU.

Cause

The aggregate method of Metric.java returns an aggregated metric from a series of input metrics and associated aggregation weights. The aggregation of verification results is handled by the aggregate method of MetricResult.java, which is called from the aggregate method of Metric.java. However, subclasses of Metric.java, such as ThresholdMetricStore.java, require aggregation of verification thresholds, as well as verification results. Thus, the aggregation method of Metric.java must be overridden by subclasses for which the aggregation of inputs involves more than simply the aggregation of metric results.

Fix

Implemented an overriding method aggregate in ThresholdMetricStore.java, which calls the superclass method to aggregate the metric results, and then aggregates the verification thresholds associated with the ThresholdMetricStore.java, together with the threshold parameter, DoubleProcedureArrayParameter.java. The verification thresholds contained in the DoubleProcedureArrayParameter.java are used by getThresholdMetadata method in ThresholdMetricStore.java to return the correct metadata for the (aggregated) Metric.java.

Notes

Examined the metadata outputs for several aggregation cases. The metadata contained the correct sample counts and thresholds for the aggregated results.

Fix: Failure to clear target attribute units when current units are absent [BUG]

Description

The original and target attribute units of the forecasts and observations are displayed in the Additional options dialog, which is accessible through 2b. Identify input data

sources > More. Target attribute units can only be defined alongside existing attribute units. However, the table entry allows for the target attribute units to be displayed when the existing attribute units are undefined.

Cause

Failure to clear the target attribute units or prevent input of target attribute units until the existing attribute units are defined. Specifically, the setValueAt method of the ScaleTableModel.java accepts and displays target attribute units when the existing attribute units are undefined.

Fix

Updated the setValueAt method of ScaleTableModel.java to only accept target attribute units when the existing attribute units are defined and to clear the target attribute units when the existing units are cleared.

Notes

Tested the table entries for the existing and target attribute units of the forecasts and observations. The target attribute units are only displayed when the existing attribute units are defined, otherwise the target units are cleared.

Fix: Failure to read forecast issue time from NetCDF forecast files [BUG]

Description

When processing NetCDF forecast files, the forecast lead times were not processed correctly unless the first entry in the time-series was an analysis/issue time (T0). For example, when the first entry comprised a forecast time, the lead time associated with that forecast was assigned zero, because the first entry was assumed to contain an analysis time.

Cause

The NetCDFEnsembleHandler.java overrides the setTimeInfo method of EnsembleDataParserCallback.java, which sets the time information associated with the forecast time-series. Forecast lead times were determined by subtracting the forecast valid time associated with each entry from the valid time associated with the first entry, which was assumed to contain the analysis/issue time. This assumption is not appropriate, as NetCDF files are not required to contain an analysis time in the first entry.

Fix

Updated the setTimeInfo method of the EnsembleDataParserCallback.java to accept a forecast start time and updated the overridden method in NetCDFEnsembleHandler.java to compute the forecast valid times relative to the specified start time. Updated NetcdfUtils.java to include a static variable identifier, FORECAST_T0="analysis_time", which is used to identify the forecast issue time in the NetCDF file. Updated the SimpleEnsembleNetcdfDataReader.java to read the issue time from a NetCDF forecast file and throw an exception if the variable "analysis_time" cannot be located.

Notes

Tested the updated handlers using a NetCDF ensemble forecast file with an issue time defined in an analysis_time variable. The file was processed correctly. However, the processing of NetCDF files remains experimental.

Fix: Failure to aggregate observations to forecast scale [BUG]

Description

Forecasts and observations are paired at a common temporal scale. When the forecasts and observations comprise different temporal scales, one dataset must be aggregated (since disaggregation is not supported). When attempting to aggregate observations, an exception was thrown, indicating that the *forecasts* could not be *disaggregated*.

Cause

The computeAndSetPairs method of the PairedDataSource.java determines separately whether the observations or forecasts require a change of scale. A change of scale in one is mutually exclusive of a change of scale in the other. However, an if/else design was not used in determining whether to aggregate the forecasts or observations. Following aggregation of the observations, a separate test was applied to the forecasts, which resulted in an exception (indicating that the forecasts could not be disaggregated).

Fix

Updated the computeAndSetPairs method of PairedDataSource.java to use an if/else design in testing whether to aggregate the observations or forecasts.

Notes

Tested the updated method with an example that required aggregation of the observations, as well as an example that required aggregation of the forecasts. In both cases, the appropriate data was aggregated, without an exception being thrown.

Fix: Inefficient saving of local data in the EVS GUI [BUG]

Description

The second window of the Verification stage saves local data upon entry in the EVS GUI. The local data is only saved to an EVS project file upon an explicit request to save the EVS project. A listener is registered to the threshold table for each verification metric, which saves any edits to the thresholds in the local data store. Upon editing a given cell in the table, the listener was triggered for *every* cell in the table, rather than once for the edited cell.

Cause

The showLocalData method of VerificationB.java registered a CellEditorListener for each cell in the threshold table for a given verification metric. Upon editing a table cell, the listener fired a ChangeEvent for every cell in the table, which called the saveLocalData method of VerificationB.java. Rather than associating a listener with each cell, a listener should be associated with each column in the threshold table.

Fix

Updated the showLocalData method of VerificationB.java to avoid registering a CellEditorListener with each cell in the threshold table. Updated the setTables method of VerificationB.java to register a CellEditorListener for each column class in the threshold table, which calls the saveLocalData method once for each edit.

Notes

Added a print statement to the saveLocalData method of VerificationB.java to identify the number of calls to this method before and after the updates. After the updates, a single edit resulted in a single call to saveLocalData, which is the expected behavior.

Fix: Incorrect updating of local data in the EVS GUI [BUG]

Description

Local data is saved in each window of the EVS GUI upon entry for the active VU. When selecting a metric in the second window of the Verification stage and updating

the local data for that metric, the updates were applied to the corresponding metric for all VUs in the active project, rather than the active VU only.

Cause

The saveLocalData method of GUIInterface.java is implemented by VerificationB.java and other windows in the EVS GUI. Rather than obtaining the active VU from an input variable, the saveLocalData method must call the getSelectedUnit method of VerificationA.java. However, depending on the context in which saveLocalData is called, updates may be required to a VU other than the active VU returned by the getSelectedUnit method. For example, a ListSelectionListener is registered with the table of VUs in VerificationA.java. Upon selecting a new VU, the updateUnitSelection method of VerificationA.java calls the saveLocalData method of VerificationB.java. This call is made only when the ChangeEvent indicates that the value is adjusting. Under these conditions, the getSelectedUnit method returns the currently selected VU, and not the previously selected VU for which the local data must be saved. Rather than obtaining the appropriate VU indirectly, the saveLocalData method of GUIInterface.java should obtain this VU as an input variable. Similarly, the showLocalData method of GUIInterface.java should obtain the appropriate VU as an input variable.

Fix

Updated the saveLocalData and showLocalData methods of the GUIInterface.java and all implementing classes to include the VU for which any changes are required as an input variable, avoiding the need to obtain this VU indirectly.

Notes

Upon selecting a metric in second window of the Verification stage and updating the local data for that metric, the updates were no longer applied to the corresponding metric associated with other VUs in the active project.

Fix: Failure to conduct spatial aggregation for probability thresholds [BUG]

Description

When forming an AU by averaging the verification results across several VUs, the verification results for a ThresholdMetric should be averaged together with the verification thresholds. When computing the VUs and AUs together, both were computed correctly. However, when separating these stages (i.e. computing the VUs

first and then computing the AU), an exception was thrown for each ThresholdMetric, indicating that the verification thresholds could not be aggregated.

Cause

The aggregate method of DoubleProcedureParameter.java computes an aggregate verification threshold from a vector of input thresholds (instances of DoubleProcedureParameter) using a specified aggregation function and corresponding vector of weights (one per threshold). Alongside the threshold value, each input threshold comprises several logical parameters. These parameters should be maintained in the aggregated output. In practice, however, they were not maintained. This resulted in a conflict in the aggregated DoubleProcedureParameter and, ultimately, an exception upon attempting to aggregate the input thresholds.

Fix

Updated the aggregate method of DoubleProcedureParameter.java to ensure that the aggregated verification threshold comprised the same logical parameters as the input thresholds.

Notes

Reproduced the conditions that led to the exception being thrown, and the thresholds were computed correctly, without exception.

Fix: Support association of EVS project files with the EVS application [ENHANCE]

Description

In order to support the association of EVS project files with the EVS application, an explicit command line option is required to instruct the EVS GUI to open with a specified EVS project file, rather than running the EVS project silently (default behavior).

Cause

In starting the EVS application, the EVS main method in EVSMainWindow.java was unable to handle a request to open the EVS GUI with a specified EVS project file, instead interpreting this as a request to run the EVS project file silently.

Fix

Updated the main method of EVSMainWindow.java to accept a “–gui” option. This opens the EVS GUI and attempts to load an EVS project file, which is specified after the –gui argument. The –gui option allows for EVS project files to be associated with

the EVS application on a given O/S. For example, in a Windows environment, the use of a batch file with the variable substitution, %1, allows for an EVS project file to be opened in the EVS GUI by double-clicking on the EVS project file. In this scenario, the batch file calls the EVS GUI and passes the EVS project file to the main method of EVSMainWindow.java using the %1 variable substitution, i.e. –gui %1.

Notes

Tested the default file association in Windows 7. Created a batch file with a single argument: java –jar EVS.jar –gui %1. Edited the registry to associate the .evs file extension with the newly created batch file. Double-clicked on an EVS project file, which was opened automatically in the EVS GUI.

Fix: Abstraction of chart coloring constraints to class constructors [ENHANCE]

Description

Constraints on the color options for lines in the default verification plots were previously applied upon adding each new dataset to that plot. Specifically, there was a constraint on the yellow color option, which was visually indistinct from the default, white, background. This constraint was applied in the addDataset method of each default chart, namely DefaultXYPlot.java, DefaultXYPlotByLeadTime.java and DefaultXYAndSamplePlot.Java. Rather than applying this constraint for each new dataset, a more efficient design would constrain the options in the class constructor.

Cause

Incorrect level of abstraction of the constraints on charting colors. These constraints were previously applied when adding a new dataset to each chart, rather than on chart construction.

Fix

Abstracted the constraints on chart colors, notably on the use of yellow, from the addDataset method of DefaultXYPlot.java, DefaultXYAndSamplePlot.Java, and DefaultXYPlotByLeadTime.java, to the constructors of these classes.

Notes

Compared the charts produced by each default plotting class before and after the change. The outputs were consistent.

Fix: Placed the advanced options for aggregating pairs in a new window [ENHANCE]

Description

The advanced options for dealing with global constraints on the verification pairs, such as subsets of dates, forecast and observed values, and aggregation and other options, are accessible from 2c. Set time parameters > More. One of the windows, Other options, contained a large number of constraints on aggregating pairs, together with a single, unrelated, option on the minimum (fractional) sample size required for metrics to be computed at a given forecast lead time. A more intuitive separation would abstract the aggregation options into a separate window.

Cause

Mixing of aggregation and other options in the Other options window of the advanced options for setting global constraints on the paired data.

Fix

Created a separate window for the aggregation options and retained the remaining, unrelated, option in the Other options window.

Notes

Tested the operation of the GUI with the new window for aggregation options.

Fix: Implemented back-end support for rolling aggregations [ENHANCE]

Description

Previously, aggregation of the verification pairs did not support a rolling aggregation, only a back-to-back aggregation of the input time-series. Examples of rolling aggregations include a 3-day moving average (aggregation period) computed every day (aggregation interval), and a 10-day accumulation computed every month.

Cause

The getTimeAggData methods of PairedData.java only supported a back-to-back aggregation of the input time-series and not a rolling aggregation with a specified interval or frequency between aggregations.

Fix

Extended the getTimeAggData methods of PairedData.java to support rolling aggregations of input time-series (observations, forecasts or pairs) with a specified interval between aggregations. Added a new class to store the aggregation

parameters, VerificationResolution.java. Updated ProjectFileIO.java, VerificationUnit.java and all method calls to getTimeAggData to use the new class.

Notes

Conducted extensive testing of various scenarios for rolling aggregations, such as 1, 3-, and 5-day means computed every day from 6-hourly and 24-hourly input data. Tested reading and writing of EVS project files with the new parameters for rolling aggregations. The files were written and read successfully.

Fix: Implemented front-end support for rolling aggregations [ENHANCE]

Description

Implemented front-end support for rolling aggregations in the GUI.

Cause

The need to include options for rolling aggregations in the Aggregation options window of the advanced options for 2c. Set time parameters.

Fix

Added options to support rolling aggregations in MoreVerificationWindowDialog.java, which is accessible from 2c. Set time parameters > More. The rolling aggregation options were added to the Aggregation window, including a drop down menu for the aggregation type {BACK-TO-BACK, ROLLING}, for which the default is BACK-TO-BACK. When selecting a ROLLING aggregation, input options appear for the interval or frequency of the aggregation and the associated temporal units.

Notes

Tested the input options for rolling aggregations, including the correct display of options stored within an existing EVS project file and the correct setting, saving, and writing of options.

Fix: Allowed conditions on valid dates to select whole forecasts [ENHANCE]

Description

When applying global conditions to subset pairs based on forecast valid date, those conditions were applied strictly to each forecast valid date/time, selecting only those forecasts (pairs) that met the conditions on valid date. Thus, forecast time-series (traces) were curtailed when one or more valid times fell outside the date constraints.

A soft constraint would allow for the selection of whole forecasts; that is, forecasts for which any of the valid dates and times meet the constraints imposed.

Cause

The inability to select whole forecasts when applying a condition on forecast valid date. Specifically, the inability of the DateCondition.java class to discriminate between individual forecasts (hard constraint on valid date) and whole forecasts (soft constraint on valid date).

Fix

Updated DateConditions.java to support either a hard constraint or a soft constraint on forecast valid date and time. A hard constrain selects only those individual forecast valid dates and times that meet the constraint. A soft constraint selects whole forecasts for which one or more valid dates and times meet the constraint. Also, updated the EVSProjectFileIO.java to support the identification of a soft versus hard constraint (one additional parameter) and added an input option to the Other options window of 2c. Set time parameters in the GUI.

Notes

Tested the soft constraint on forecast valid date and time, together with the associated project file I/O and the graphical input and display of this soft constraint.

Fix: Added option in GUI for setting verification window in valid time [ENHANCE]

Description

The EVS project file supports an advanced option to set the time system of the verification window (i.e. the start and end dates of the period to consider for verification). The options include issue/basis time or valid time, and the default option is valid time. This option was only accessible from the EVS project file, and was not supported in the GUI.

Cause

Lack of front-end support in the GUI to set the time system for the verification window.

Fix

Updated MoreVerificationWindowDialog.java to include a checkbox for setting the time system of the verification window to valid time or issue/basis time, where the

default is valid time (checked). The new option is accessible via 2c. Set time parameters > More > Other options.

Notes

Tested the front-end support for setting the time system of the verification window. The option was displayed, saved locally, and written/read correctly to/from the EVS project file.

Fix: Implemented front-end support for identifying inadmissible data [ENHANCE]

Description

The EVS project file supports an advanced option for defining inadmissible data. Inadmissible data is identified with a threshold, an associated logical condition (e.g. less than), and a value to assign for data that meet the condition. The condition applies to both forecasts and observations. For example, a lower bound of 0.25mm may be defined for precipitation, with values below 0.25mm assumed to represent model/measurement uncertainty and, therefore, assigned zero. Previously, this option was not supported in the GUI.

Cause

Lack of front-end support in the GUI to impose constraints on admissible data values.

Fix

Updated MoreInputDialog.java to include inputs for the threshold and logical condition that define inadmissible data and the value to assign for inadmissible data. These options are accessible via 2c. Set time parameters > More > Other options.

Notes

Tested the front-end support for identifying inadmissible data values. The options were displayed, saved locally, and written/read correctly to/from the EVS project file.

Fix: Added a metric comprising modified box plots by forecast value [ENHANCE]

Description

A new metric comprising modified box plots by forecast value was added to the EVS. The forecast values are determined by applying a specified function (mean, median or mode) to the ensemble members. When several forecasts are identified with a common forecast value, the errors are pooled across those forecasts.

Cause

Inability to plot modified box plots and order them by forecast value.

Fix

The new metric is implemented in ModifiedBoxPlotsByLeadFcst.java, which extends a new superclass, ModifiedBoxPlotByLeadVal.java. The superclass allows for box plots to be organized by observed or forecast value. The ModifiedBoxPlotByLeadObs.java was also modified to implement the new superclass. Added a new metric parameter, BoxUnpooledFcstPointsParameter.java, to store the number of equally-spaced quantiles at which the boxes are computed and drawn. Updated VerificationB.java to provide front-end support and updated ProjectFileIO.java to allow for saving of the metric configuration to an EVS project file. Implemented a default plotting class, ModifiedBoxPlotUnpooledByLeadFcst.java, and associated superclass, ModifiedBoxPlotUnpooledByLeadVal.java. The ModifiedBoxPlotUnpooledByLeadObs.java was also updated to implement the new superclass.

Notes

Tested the new metric for several verification projects, displaying the plots within the EVS GUI, writing the graphical and numerical outputs, saving the configuration to a project file and re-reading the configuration.

Fix: Improved the handling of missing values [ENHANCE]

Description

Missing values were handled through a global no-data value that applied to all input data sources, as well as internal calculations and outputs (e.g. of the verification pairs). The global no-data value is accessed through 2b. Identify input data sources > More > Other options > Global no-data value. For those input data sources that contained explicit missing value identifiers (e.g. PI-XML), these identifiers were ignored and any missing values interpreted according to the global no-data value. As such, separate missing value identifiers were not supported for observed and forecast data sources. Further, NaN was not supported as a missing value identifier.

Cause

The inability to handle explicit identifiers for missing values in the observed and forecast input data sources, together with the inability to handle NaN as a missing value. For those files that contain missing value identifiers, the missing values should

be replaced, via the appropriate data handlers, to the global no-data value. These data handlers should also accept (and replace) NaN as a missing value.

Fix

Updated the data handlers for PI-XML, Fast Infoset, and NETCDF to identify missing values according to the explicit identifiers provided. Any missing values (included NaN values) are now replaced with the global no-data value. As before, the global no-data value is used to discriminate between valid and missing data internally and in the EVS outputs (e.g. verification pairs). Likewise, for those data sources that do not explicitly define a missing value, any values that match the global no-data value are interpreted as missing. By default, the no-data value is -999.0 and only numeric missing values are accepted. However, as indicated above, input data sources with a missing value of NaN (or a prescribed numeric value) are now replaced with the global no-data value.

Notes

Tested the modified data handlers with input data sources that contained missing values, including NaN. The missing values were converted to the global no-data value, as expected.

Fix: Added an ensemble ID to discriminate between input time-series [ENHANCE]

Description

Forecast files in the PI-XML and FI/BIN formats may contain several time-series for any given location (locationID) and variable (parameterID). These time-series are discriminated with an ensembleID. However, the corresponding EVS data handlers were unable to discriminate by ensembleID, instead using the locationID and parameterID alone. Thus, files containing multiple time-series for a given locationID and parameterID were not processed correctly.

Cause

For data in PI-XML and FI-BIN format, the inability to discriminate between time-series with a common locationID and parameterID using their unique ensembleID.

Fix

Enhanced the data handlers for PI-XML and FI/BIN to optionally discriminate time-series on the basis of ensembleID, as well as locationID and parameterID. The ensembleID is only required when the input file contains more than one time-series with the same locationID and parameterID. Updated the MoreInputDialog.java to

include an editor for the ensembleID in the GUI, which is accessed from 2b. Identify input data sources > More > Other options > Forecast file ensemble ID.

Notes

Tested the modified data handlers for time-series that were discriminated by ensembleID. For those files containing more than one time-series with the same locationID and parameterID, an error was thrown in the absence of an ensembleID, requesting input of the ensembleID. When the ensembleID was defined, the input time-series were discriminated and processed correctly. While the EVS now supports time-series in PI-XML and FI-BIN format that are discriminated by ensembleID, it is recommended that large datasets are distributed across several files, in order to increase flexibility (i.e. to consider only required data) and to minimize I/O time, which is generally a significant fraction of the total time required for verification.

4.0 Changes from EVS 5.1 to EVS 5.2

Fix: Incorrect addition of default Aggregation Unit (AUs) [BUG]

Description

In the Aggregation window of the GUI, copying a default Aggregation Unit (AU) and then attempting to delete that AU, failed to remove the selected AU. Specifically, upon deleting the default AU, a new default AU was immediately added to the list of AUs. Thus, only fully-defined AUs could be deleted from the list.

Cause

In order to avoid the addition of duplicate AUs, the addDefaultUnits method of AggregationA.java checks for existing AUs with common parameter values. However, it was only checking for fully-defined AUs that would lead to duplication, and not for default AUs with the same candidate Verification Units (VUs).

Fix

Fixed the addDefaultUnits method of AggregationA.java to check for default AUs with the same candidate VUs before adding new default AUs.

Notes

Activated the same code by copying a default AU and then deleting the AU. This resulted in the AU being removed correctly.

Fix: Misleading error message on setting incorrect weights [BUG]

Description

Weighted aggregation of two of more VUs requires the weights associated with each VU to be defined in the Aggregation window. The weights must sum to 1.0 and an error message is encountered when the weights do not sum to 1.0. However, when an AU has not been previously saved, the error hierarchy leads to a misleading, generic, error being thrown, which fails to identify the lack of unity in the weights as the trigger.

Cause

The catch block of the saveData method in AggregationA.java catches an error in the weights but, before throwing the error, attempts to reset the original weights. Since this attempt can fail, a more generic error may be thrown, triggering an exit of the saveData method before the original error can be thrown.

Fix

The catch block of the saveData method was updated. The code that attempts to reset the default weights is now placed in a try/catch block. If this fails, the error is caught, and the original error (highlighting weights that do not sum to unity) is correctly thrown. A failure to reset the original weights may occur if no valid weights were previously defined.

Notes

Setting incorrect weights on first saving an AU now results in the correct error message being thrown, instead of a more generic message, indicating to the user that the problem originates in the weights not summing to unity.

Fix: Failure to save a manually entered forecast file data source [BUG]

Description

When attempting to save a manually entered path to a forecast file data source, the path was not properly saved unless an earlier path existed. In contrast, when an earlier path existed or the file dialog was used to populate the input, the path was properly saved.

Cause

An if-clause in the saveLocalData method of VerificationA.java was poorly composed. This resulted in a failure to catch and handle the above scenario (manually entered forecast file data source, without a prior data source defined), namely by saving the manually entered path to the forecast file data source.

Fix

Updated the if-clause in VerificationA.java to properly handle the scenario where a forecast file data source is manually defined for the first time.

Notes

Checked all possible combinations for defining the forecast file data source.

Fix: Failure to re-populate the list of a searchable combo-box [BUG]

Description

A searchable combo-box allows for a list of items in the combo box to be re-populated with matching search text. However, upon finding an exact match, the list was re-populated with a single item and the search text needed to be deleted in order

to restore the full (original) list. Also, when finding no matches, the combo box was populated with an empty list, rather than providing the full original list.

Cause

Two bugs were found in the SearchableComboBox.java, the first concerned with an exact match (list with one item) and the second concerned with an empty list (no matches). In both cases, the list associated with the combo-box was not being re-populated with the original items.

Fix

Corrected two bugs in the SearchableComboBox.java to ensure the combo-box list is re-populated with the original list when either a single item is found or no items are found.

Notes

Manually checked all keyboard and mouse interactions with the SearchableComboBox to ensure that: 1) the full list is returned when a single item is found or no items are found from the search text; and 2) a reduced list (based on the matching text) is returned in all other cases.

Fix: Failure to warn of permission errors when saving a project file [BUG]

Description

When encountering a file permission error on writing an EVS project file, the resulting IOException was output to the command line only and failed to display in the GUI exception handler.

Cause

The writeObjects method of the EVSMainWindow.java class failed to throw an IOException, instead catching all exceptions and printing to standard out.

Fix

The writeObjects method of the EVSMainWindow.java class was modified to propagate all IOExceptions upwards, allowing for proper handling by calling methods, such as displaying the IOException in the GUI exception handler.

Notes

Recreated the problem by changing the file permission on an EVS project file to read only. Prior to the bug fix, any attempt to save the EVS project file would result in a

"silent" error, printed to standard out only. Following the fix, the IOException is properly displayed in the GUI exception handler.

Fix: Disappearing modal dialogs [BUG]

Description

Under some (inconsistent) conditions, a modal dialog would disappear behind other open windows, blocking further input to the EVS until the modal dialog re-appeared through trial-and-error minimization/maximization of other open windows.

Cause

The parent frame of the modal dialog was not being set properly on construction. Specifically, due to the order of construction of the modal dialogs before the main EVS main window, a null parent frame was passed to the constructors of the modal dialogs.

Fix

Changed the order of construction of the modal dialogs to follow the construction of the EVS main window. The modal dialogs are now constructed with a non-null parent frame.

Notes

Tested the modal dialogs under Windows and Linux. No further blocking of modal dialogs occurred.

Fix: Incorrect re-weighting of VUs associated with an AU [BUG]

Description

Upon changing the weights of a VU associated with an AU in AggregationA.java, the remaining weights are updated accordingly (e.g. to re-assign equal weights when VUs are added or removed). The updates are coordinated by the setValueAt method of the table model associated with the table of VUs, which is called upon a change in the selection of a VU or a change in weight. However, the checks were not conditional upon the VU being flagged for inclusion or on the updated value of the entry passed to setValueAt. This led to incorrect updates to the weights upon editing unchecked VUs or when the current VU was being deselected (because the status was checked against the existing selection, not the updated selection).

Cause

Failure to properly check the status of the VUs and their associated weights in the setValueAt method of the table model associated with the table of VUs in AggregationA.java.

Fix

Updated the setValueAt method of AggregationA.java to properly check the status of the VUs and their associated weights upon adding or removing VUs or changing weights.

Notes

Tested the re-weighting of VUs upon adding or removing VUs and when the edited weight belongs to a VU that is checked versus unchecked. All scenarios performed as expected.

Fix: Failure to fully clear and reset the GUI on creating a new EVS project [BUG]

Description

On creating a new EVS project, the GUI was not fully cleared, potentially leaving results from a prior open project. In addition, the GUI failed to return to the first window in the verification stage.

Cause

The newProject method of EVSMainWindow.java failed to clear all prior data and reset the GUI to the first window of the verification stage.

Fix

Updated the newProject method of EVSMainWindow.java to clear all prior data and reset the GUI to the first window of the verification stage.

Notes

Tested by generating results for one EVS project and then creating a new EVS project. On creating the new project, all data from the old project were fully cleared and the GUI returned to the first window in the verification stage.

Fix: Extend list of observed file types displayed in file dialog [ENHANCE]

Description

By default, only selected file extensions were identified as “observed” data files in the file dialog. The default file extensions did not include .MAP06 or .MAT, for example. In order to display these files in the dialog, it was necessary to select “All Files” in the drop-down menu provided in the file dialog. This was potentially confusing, as it may not be apparent that the file dialog is showing only a subset of the available files, based on the default extensions.

Cause

Limited list of default file extensions for observed data files. In particular, the default file types did not include MAT, MAP06 and MAP01, which are relatively common.

Fix

Added .MAP06, .MAP01, and .MAT to the list of default file types in the file_types.xml file of the nonsrc/parameterfiles directory.

Notes

Checked that the additional default file types were correctly displayed in the file dialog when choosing observed data files.

Fix: Handling of aggregation weights in the GUI [ENHANCE]

Description

When setting the weights for spatial aggregation of several VUs, there were several minor constraints in the aggregation window of the GUI that interrupted smooth user interaction.

Cause

When editing the weight for one VU in an AU comprising only two units, the weight of the remaining unit was not automatically updated, based on the requirement for the total weights to sum to 1.0. In addition, the editing cell in the table was not automatically closed upon completion, leaving the editor open when selecting other VUs. Finally, when the weights did not sum to 1.0, the error message was unclear about which AUs (in the presence of multiple units) failed to sum to 1.0.

Fix

Updated AggregationA.java to allow for the second of a pair of weights to be updated automatically when the AU comprises only two VUs and the first of the pair of weights is edited. The automatic update ensures that the weights sum to 1.0, as required. Enforced closure of all open cell editors in the table of weights in AggregationA.java when switching between VUs or saving the project. Improved the warning message when weights do not sum to 1.0, identifying the first AU for which the weights are incorrect.

Notes

Checked all enhancements manually by interacting with the GUI for different scenarios of the aggregation weights.

Fix: Support reading and writing of EVS pairs in compressed format [ENHANCE]

Description

The EVS pairs are written in an uncompressed ASCII XML format. In order to conserve disk space, the option for writing (and reading) of the EVS pairs in a compressed (gzip) format is useful.

Cause

Lack of functionality for writing and reading of the EVS paired files in a compressed (gzip) format leads to inefficient use of disk space.

Fix

Added optional functionality to read and write the EVS paired files (both raw and conditional) in a compressed gzip format. The default remains to write the pairs in an uncompressed format. The option to write the pairs in a compressed gzip format is accessed through the output options dialog in the first verification window. The output options dialog is accessed via the “More” button in “2d. Select location for output data”. Updated the I/O of the EVS project file to store the preference for compressed output. The option appears under the <write_gzip_pairs> tag of the <paired_data> block) and applies to each VU separately.

Notes

Checked that the compressed files were written and read correctly and that the option was written and read correctly to/from the EVS project file.

Fix: Dismissing multiple warnings with a single action [ENHANCE]

Description

Upon deleting multiple VUs, several warning messages may be thrown if there are dependencies between VUs. Rather than scrolling through all messages, the option to ignore all warnings would be useful.

Cause

The inability to accept all warnings when deleting multiple VUs.

Fix

Added an option in VerificationA.java to accept all warnings when deleting multiple VUs that have dependencies with other VUs (not all of which may be deleted). Each warning dialog now comprises an option “Yes to all” to accept all subsequent warnings and proceed.

Notes

Manually tested all three options, namely to cancel the delete, accept the current warning message and proceed (possibly to other warning messages), and to accept all warning messages and proceed to delete. All options worked as anticipated.

Fix: Support reading of Fast Infoset binary XML files [ENHANCE]

Description

Fast Infoset XML provides a binary interface to XML that allows for smaller file sizes and more efficient processing times.

Cause

Inability to process Fast Infoset binary XML files for the ensemble forecasts and observations.

Fix

Created a new reader, FastInfosetXMLIO.java, to support reading of Fast Infoset binary PI-XML for both ensemble forecasts and observations. The new reader relies on library methods in ohdcommonchps.jar and two Deltares libraries, Delft_Util.jar and Delft-PI.jar.

Notes

Conducted performance testing of read times for ASCII PI-XML and Fast Infoset binary XML. The read times and disk-space footprint for the Fast Infoset binary XML files were significantly smaller than ASCII PI-XML.

Fix: Support file filter when reading forecast directory [ENHANCE]

Description

When specifying a directory from which to read forecast files, any non-forecast files within that directory are processed as forecast files, which leads to an I/O exception. This is expected behavior, but limiting when the forecasts are archived in directories with mixed files or subdirectories.

Cause

Inability to process a directory containing mixed forecast files by listing and processing only those files that match a prescribed filter.

Fix

Implemented a file filter to screen a directory of forecast files and process only those files that match a prescribed pattern. This only applies when a directory is selected as the forecast data source, and the filter must be defined explicitly (the default is to no filter). Added a new class, Filter.java, to filter the files, and updated the FileDataSource.java to store the filter. Also updated the ProjectFileIO.java to support reading/writing of the new filter option and MoreInputDialog.java to allow the filter to be entered in the GUI. Finally, VerificationA.java was updated to accommodate saving of the filter information.

Notes

Tested the default behavior (no filter) and the new filter option. The default behavior was maintained and the filter worked as expected, screening only those files that matched a prescribed pattern. The filter option was preserved in the GUI and in the project file, as expected.

5.0 Changes from EVS 5.0 to EVS 5.1

Fix: Misleading display of reference forecasts for aggregation units (AUs) [BUG]

Description

When aggregating verification results from several locations, multiple Verification Unit (VUs) are used. Each VU has a separate reference forecast for computing skill and the overall results are provided in an Aggregation Unit (AU). In the EVS verification plots, the reference forecast is included in the plot title for skill metrics. For an AU, the reference forecast is shown for the first VU that forms the aggregation. However, this was not qualified in the plot titles as being the reference forecast associated with the "first" VU only.

Cause

For AUs, a failure to identify the reference forecast in the plot titles for skill scores as being the reference forecast associated with the "first" VU that forms the aggregation.

Fix

Qualified the EVS plot titles for aggregated verification results. The reference forecast is now qualified as being the "first" of several potential reference forecasts (one for each VU forming the aggregation).

Fix: Bug in the getWidestItemWidth method of AdaptableComboBox.java [BUG]

Description

The getWidestItemWidth method of AdaptableComboBox.java sets the width of the AdaptableComboBox dynamically depending on the width of the widest element in the box. However, the box allowed for iteration over null elements, which resulted in a NullPointerException when the box comprised no elements.

Cause

Failure to screen out null elements when setting the width of the AdaptableComboBox dynamically.

Fix

Added a check for null elements in the getWidestItemWidth method of AdaptableComboBox.java

Fix: Bug in the setCellEditors method of EditorTable.java [BUG]

Description

The setCellEditors method of the EditorTable.java accepted generic objects as cell editors, while the internal code of EditorTable.java casts to TableCellEditor.java objects in various places, potentially resulting in a ClassCastException.

Cause

Failure of setCellEditors to take TableCellEditor.java objects.

Fix

Updated the setCellEditors method of EditorTable.java to take objects of class TableCellEditors.java only.

Fix: Bug in modality of two GUI classes [BUG]

Description

The MoreInputDialog.java and the MoreVerificationWindowDialog.java did not previously allow for modality on construction. However, parameters in these dialogs are affected by choices in other dialogs. Thus, modality is required to ensure that the dialogs are always in a consistent state, i.e. that entries cannot be changed elsewhere before closing these dialogs.

Cause

Inability to set the modality of MoreVerificationWindowDialog.java and MoreInputDialog.java and to ensure the default setting for each dialog is modal (to prevent an inconsistent state arising from changes made elsewhere).

Fix

Enhanced the constructors of MoreVerificationWindowDialog.java and MoreInputDialog.java to allow for modality. Changed the default setting to modal.

Fix: NullPointerException in SearchableComboBox.java [BUG]

Description

In displaying a string representation of items in the SearchableComboBox, toString was potentially called on null items, resulting in a NullPointerException.

Cause

Failure to check for null items in the SearchableComboBox before calling toString on those items.

Fix

Added checks for null data items, such that toString is only called on non-null data items.

Fix: Spelling error in the html explanation of the Brier Skill Score [BUG]

Description

Spelling errors were identified in the html description file for the Brier Skill Score (bss.html), which is displayed in the table of verification metrics in VerificationB.java.

Cause

Spelling errors in the html description file.

Fix

Corrected the spelling errors.

Fix: Real-valued verification thresholds not aggregated properly [BUG]

Description

When aggregating verification metrics by averaging across the inputs, the real-values associated with the probability thresholds were not previously aggregated. Rather, the nominal value of the output threshold was equal to the real value associated with the first input. For consistency with the aggregation procedure applied to the metric values, the same aggregation function is now applied to the input thresholds, and the output threshold comprises an aggregate of the input thresholds. This is a nominal value only, since the actual verification is done separately for each input.

Cause

Failure to aggregate the real-valued thresholds associated with an AU when conducting aggregation by averaging metrics.

Fix

Added a method, aggregate, to DoubleProcedureParameter.java. The method conducts averaging of the real-valued thresholds associated with multiple VUs when forming an average of the inputs metrics. Updated the aggregate method of

MetricResultByThreshold.java, which controls the aggregation across multiple metrics, in order to call the new aggregate method in DoubleProcedureParameter.java.

Fix: Incorrect re-scaling of aggregation weights when missing first metric [BUG]

Description

Failure to correctly re-scale the weights associated with multiple VUs when aggregating across missing metrics and, specifically, when the first metric is missing. For example, when aggregating metrics across three basins {A,B,C} with weights of {1/3,1/3,1/3}, a missing metric for basin A {MISSING, B, C} resulted in an aggregation of 1/3*B+1/3*C instead of 1/2*B+1/2*C.

Cause

Failure to rescale aggregation weights properly in the aggregate method of the MetricResultByThreshold.java class when the first metric is missing.

Fix

Fixed the aggregate method of the MetricResultByThreshold.java class to correctly rescale weights when the first of several metrics is missing.

Fix: Changed default behavior for "reset" buttons [BUG]

Description

Previously, the default behaviour for the "reset" buttons in the GUI dialog evs.gui.windows.MoreInputDialog.java was to clear all contents in the associated tables rather than reset the contents to their original state.

Cause

The default behavior of the "reset" buttons in the GUI dialog evs.gui.windows.MoreInputDialog.java was misleading, clearing the contents of the associated tables, rather than resetting the contents to the original state.

Fix

Updated the default behavior of the methods associated with the "reset" evs.gui.windows.MoreInputDialog.java to return the contents of the tables to their original state, rather than simply clearing contents.

Fix: Renamed MoreOutputOptions.java [ENHANCE]

Description

The naming convention for EVS advanced dialogs is to append “dialog” to the class name. Thus, renamed MoreOutputOptions.java to MoreOutputOptionsDialog.java for consistency.

Cause

Inconsistent naming of MoreOutputOptions.java.

Fix

Renamed MoreOutputOptions.java to MoreOutputOptionsDialog.java. The refactoring was conducted safely.

Fix: Added functionality to allow for joint pairing [ENHANCE]

Description

When comparing verification metrics from two forecasting systems, whether informally or using an explicit measure of skill, it is generally preferred to use common forecast valid dates and times. In order to support this, a mechanism was implemented to jointly pair two VUs; that is, to select only the common pairs from both VUs. For example, if the pairs used in the denominator of a skill score cover a different period than the pairs used in the numerator, the skill may reflect a combination of data variability (not desirable) and real differences in forecast quality (desirable).

Cause

Previously, it was not possible to coordinate two VUs and conduct verification for only those pairs that appear in both VUs.

Fix

Changes were required to several EVS classes, including GUI and non-GUI classes. In particular, updated the getMergedData method of PairedData.java to compute the intersection of two paired inputs while only returning the pairs associated with the first input (i.e. pairs that are also present in the second input). Also, renamed the getMergedData methods to getIntersection. Updated VerificationUnit.java to store the VU on which joint pairing is conducted, together with associated getter and setter methods. Also updated the getConditionalPairs method of VerificationUnit.java to

compute the conditional pairs subject to any newly defined constraint on joint pairing. Added a new method syncJointPairs to VerificationA.java to synchronize any VUs that are used as joint pairs following a change in the name of the VU. Updated the read/write methods of ProjectFileIO.java to read and write the new parameter for joint pairing, which appear in the <paired_data> block of the EVS project file. Finally, updated the MoreOutputOptionsDialog.java to include a new row for specifying a VU on which to conduct joint pairing and updated the deleteSelectedUnits method of VerificationA.java to warn on deleting a VU that is used elsewhere for joint pairing. Performed extensive testing of the new functionality.

Fix: Improved organization of MoreParOptionsDialog.java [ENHANCE]

Description

The close method of MoreParOptionsDialog.java failed to make best use of existing code elsewhere in MoreParOptionsDialog.java. By splitting close into two methods, saveParameters and close, code re-use is improved.

Cause

Failure to of close method to make best use of existing code.

Fix

Split the close method into two separate methods, saveParameters and close.

Fix: Allow real-valued bounds on computing metrics [ENHANCE]

Description

Metrics may be computed with thresholds that are defined in real units or climatological probabilities. When defining thresholds in terms of climatological probabilities, the corresponding real values are not known in advance. It is sometimes convenient to specify real-valued bounds on the thresholds for which metrics are computed, avoiding wasteful computations.

Cause

Inability to constrain the real-valued thresholds for which metrics are computed (e.g. when using climatological probabilities whose real-valued thresholds are not specified in advance).

Fix

Added real-valued bounds to constrain the computation of metrics for thresholds that fall within those bounds only. This required additions to several source files, namely DoubleProcedureArrayParameter.java, MetricParameter.java, ThresholdMetricStore.java, DoubleProcedureParameter.java. It also required a new parameter to store the bounds, DoubleIntervalParameter.java. Source files associated with the GUI and reading/writing of EVS project files were also updated to accommodate the new parameters, namely ProjectFileIO.java, VerificationB.java and MoreParOptionsDialog.java.

Fix: Detection limit on forecasts and observations [ENHANCE]

Description

In forecasting and observing some variables (e.g. precipitation), a detection limit may apply, beyond which it may be preferred to assume a lower or upper bound. For example, precipitation forecasts falling below an instrument detection limit might be assigned zero. This may avoid spurious verification results or sensitivities to choice of threshold (such as the threshold for determining Probability of Precipitation).

Cause

Inability to assign detection limits to forecasts and/or observations prior to conducting verification.

Fix

Implemented a detection limit, comprising a bound, limit and function type, in a new class, DetectionLimit.java. Any value that meets the logical test defined by the function type with respect to the limit (e.g. less than the limit) is assigned the value of bound (e.g. 0.0). A detection limit is applied jointly to the forecasts and observations and is associated with a particular VU (for which a new getter and setter method was implemented). When defined, the detection limit is used by the getConditionedPairs method of VerificationUnit.java. This method returns conditional pairs after imposing any detection limit on the forecasts and observations. If a change in measurement units is requested, the detection limit is defined in the **target** measurement units of the forecasts and observations (i.e. in the units of the conditional paired data). The detection limit is an “advanced” option, not available in the EVS GUI. Rather, it is specified in the EVS project file under the <paired_data> block using a new

<detection_limit>. The <detection_limit> comprises a <limit>, <bound>, and <type>, where the latter may include: isLess, isGreater, isLessEqual, and isGreaterEqual.

Fix: Option to exit MoreInputDialog from the Forecast Scale table [ENHANCE]

Description

Previously, there was no option to exit the MoreInputDialog.java via an "OK" button from the Forecast Scale table. Rather, it was necessary to navigate to one of the adjacent tabbed panels of the MoreInputDialog.java to exit. While the previous behavior is expected usage on first entering data, it is inconvenient on editing data, since editing may focus on the Forecast Scale table only (from which an exit sequence via a call to "OK" should be possible).

Cause

Inability to exit the MoreInputDialog.java from the tabbed panel containing the Forecast Scale table.

Fix

Added an exit ("OK") option to the tabbed panel of the MoreInputDialog.java containing the Forecast Scale table.

Fix: Separate pairing options from the "Other options" dialog [ENHANCE]

Description

The MoreInputDialog.java comprises a tabbed panel with "Other options." This tabbed panel collects together a broad range of options associated with the input data. Many of these options relate to pairing of forecasts and observations. In order to improve the organization of these options and, specifically, to separate out the pairing options, a new tabbed panel of "Pairing options" was required MoreInputDialog.java.

Cause

Lumping together of too many options in the "Other options" table of the MoreInputDialog.java, including multiple options that control the pairing of forecasts and observations.

Fix

Implemented a new “Pairing options” table in the MoreInputDialog.java and moved all options related to pairing from the “Other options” table to the new table of pairing options.

Fix: Improved control on aggregating forecasts and observations [ENHANCE]

Description

Added new functionality to the EVS pairing process to allow for improved control on a change of scale of the forecasts and observations prior to conducting pairing. Previously, the choices for aggregation (and the associated controlling variables) were shared between methods used to aggregate forecasts and observations *prior* to pairing and methods used to aggregate the pairs themselves. There was no separate control on aggregating forecasts and observations before pairing. This prevented pairing in some cases (i.e. when different parameter values were required for aggregation before pairing versus aggregation after pairing). For example, if the observed data begin at 12 UTC and comprise 6-hourly values, while the forecasts begin at 0 UTC and comprise daily values, separate control on the observations would allow the first two observations to be skipped and pairing to proceed at the aggregated daily scale. Subsequently, the pairs may be aggregated to explore verification results at the 5-daily scale.

Cause

Linking of functionality for aggregating forecast and observations *before* pairing to aggregating of pairs, i.e. shared variable values for these two scenarios. This prevented separate control of aggregation for pairing purposes, which is important when the forecasts and observations are defined at different scales (i.e. when aggregation of one or the other is essential prior to pairing).

Fix

Added new variables to the VerificationUnit class, namely the fcstAggStartHourUTC and the obsAggStartHourUTC, together with the fcstStartLeadHourUTC to control, respectively, the start time for aggregating observations, the start time for aggregating forecasts and the first forecast lead time at which to begin aggregation. Implemented corresponding getter and setter methods. Updated the EVS project file I/O to accommodate the new variables. Added a new table to the advanced options

for input data, evs.gui.windows.MoreInputDialog.java, to provide access to the new options. The new table comprises these and other options for pairing data.

6.0 Changes from EVS 4.0 to EVS 5.0

6.1 Changes in default behavior

- Altered the default functionality of the EVS when reading ASCII files to interpret the date strings in a given format without leniency. For example, 200005 was leniently processed when the required format was yyyyMMddHH, despite the missing ddHH. Due to the Java API for Calendar, this lead to an undesirable interpretation of the dates (producing wrong dates without warning) when the dates on file were corrupt, instead of failing (desirable). A non-lenient interpretation of dates is now implemented and an error thrown if the dates on file do not match the required date format.
- When deriving climatological probabilities from a specified source of observed data, the climatological probabilities are derived from the paired observations unless explicitly requested to use all available observations (from the GUI, using the advanced option: 2b. Identify input data sources > More > Other options > Use all observations (not just paired) to determine climate thresholds; and from the EVS project file using the XML tag <use_all_observations_for_climatology>). Previously, the climatological probabilities were determined from all available observations *between the start and end dates of the verification period*, without any options to control this behavior. Now, the climatological probabilities may be determined from the paired data, from all available observations, or from an arbitrary subset of observations (by controlling how date and value conditions are applied to the observed data).

6.2 Bug fixes related to Graphical User's Interface

- Updated the box plots by observed value to include the units of measurement on the domain axis (previously this was stated on the range axis only).
- In the second window of the Verification stage, clicking "Do all" to apply the thresholds for the currently selected verification metrics across all other metrics required the thresholds in the table to be finalized (i.e. the table row to have stopped editing) otherwise erroneous thresholds were reproduced across all metrics. The editing is now finalized automatically before the thresholds are copied.

- Fixed a bug in AggregationA.java whereby the Aggregation Units (AUs) became incorrectly ordered upon saving, leading to an inconsistent state in the locally-stored parameter values.
- In the deleteReferenceFcst method of VerificationA.java, a map iterator was modifying elements in a map concurrently, leading to a ConcurrentModificationException.
- In VerificationB.java, when identifying thresholds as "main" in the threshold table, a modification followed by a save was not being registered on reloading a project because the table row selection was maintained across loading projects, yet the save index was cleared, leading to a conflict between the two.
- The RankHistogramPlot.java failed to plot the rank histogram in the desired bar chart form when only one “main” threshold was requested. Instead, the bar chart format was only plotted if the threshold count was one.
- Fixed a bug that prevented the proper display of the EVS html help documents in an external browser. The solution involved decompressing the html documentation on-the-fly from the EVS.jar and then pointing the browser to the decompressed files for display. All help documentation can now be viewed on-the-fly in an external web browser.
- Week 53 was not supported as a conditioning option in the GUI (this can occur in leap years). This is now supported.
- The addDefaultUnits method of AggregationA.java failed to add an AU for each unique combination of VUs.
- Fixed a bug in OutputA.java whereby a ClassCastException was thrown upon wrongly casting Double valued lead times in the lead time table as Integer objects. This only occurred when selecting a subset of lead times for display.
- The results from all VUs were removed upon cancelling the calculation of one VU (i.e. the clean-up extended too far, removing all results).

6.3 Bug fixes not related to Graphical User's Interface

- Sample counts were not being written properly to the XML metadata files.
- Timezone offsets other than UTC (i.e. zero offset) in the PI-XML forecast files were not being processed correctly.

- Fixed a bug in method setTargetMeasurementUnitsFunc of evs.analysisunits.scale.Support, whereby a request for no change in measurement units could be coupled with a function that *did* imply a change in measurement units, leading to an undesirable change in units. Added a check to prevent this.
- When using another VU as a reference forecast in a skill calculation, any renaming of that VU was not properly reflected in the list of reference forecasts.
- When loading AUs into the GUI, the project file loaded without warning if the AUs comprised VUs that were miss-spelled. Modified a try/catch block to throw an appropriate error, resulting in the (correct) failure to load the project.
- Fixed a bug in the loading of project files whereby the display of AUs was not updated if an existing project was loaded or existing units were defined but not saved in a project.
- The readInHeader method of the evs.data.fileio.ohdfile.data.Datacard file reader incorrectly threw a NullPointerException rather than an IOException when reading a null file handle, leading to an ambiguous error message about the cause of the problem.
- Fixed a bug in the readAndSetClimatology method of PairedDataSource.java whereby user-specific IDs for the observed data other than locationID and variableID were not being checked and used if available. For the PI-XML data, several IDs may be used at any given location, with different IDs for observations and forecasts, hence the need for user-specific file IDs separate from the main IDs used for verification.
- In the ProjectFileIO.java class, relative paths were not correctly saved either for AUs or for the paired data. Relative paths can now be used for all data sources in the EVS. Paths are resolved relative to the directory in which the EVS.jar is located.
- In the ASCIIFileIO.java, date strings of different length than the specified format string were still processed (i.e. the dates were interpreted leniently), leading to unexpected dates upon reading ASCII files. A non-lenient implementation was adopted.
- Fixed a bug in the determination of verification thresholds from the climatological observations. While a file containing an extended, unconditional,

climatology could be specified in place of the paired observations, the unconditional climatology was incorrectly ignored under some circumstances.

- In VerificationUnit.java, a change of units to an unconditional climatology was applied to missing data (-999.0). Following application to the missing data, these data were no longer interpreted as missing but (incorrectly) as non-missing data with arbitrary values (depending on the requested change of units).
- Fixed a bug in the PISAXHandler.Java whereby a decimal time offset in XML was not being read and interpreted properly, leading to an (incorrect) time offset from UTC of zero. This occurred when the time system in PI file was mistakenly expressed as a decimal (e.g. -5.0 for EST). A partial fix to this bug was implemented previously, but the fix failed under specific circumstances, now corrected.
- In setting explicit forecast/observed file IDs for PI-XML etc., the check on equality of null IDs was failing, leading to existing pairs and results being deleted upon opening of an additional options dialog (that contained the IDs) after a verification run; paired data are always auto-deleted upon a change in a parameter that potentially affects the pairs. The bug/improper method invocation was located in StringUtilities.nullEquals. This method was enhanced to take an additional parameter allowing empty strings to be treated as null.
- Fixed a bug in the specification of weeks of the year on which to condition, whereby integer weeks of the year were interpreted as one week too late.
- Fixed a convoluted complex bug whereby AUs were not being deleted properly for VUs whose parameters had changed, dictating the deletion of the AU. This was due to the deletion taking place within the loop in which VUs were checked, thus allowing a VU to be associated with an AU that would later disappear due to changes in other VUs (later in the loop). This specifically occurred when the offending VU comprised an AU with only two VUs of which the earlier VU was one. Moved the check for setting AUs outside of the loop for checking whether AUs should be deleted.
- In the temporal aggregation routine getTimeAggData, the number of rows to aggregate was determined from the separation in time between two forecasts. However, when a time constraint on the processed data led to the first trace having a single time, the hour determination between forecasts was incorrect.

The bug was fixed by splitting the data into traces and finding the first one with more than two consecutive times from which the forecast frequency could be determined correctly.

- The read method in PublishedInterfaceXMLIO.Java threw a generic IOException, yet code in the PairedDataSource.java that calls this method relied on a ConditionException being thrown and caught when the file was processed properly but did not contain data of interest. The result was that, when processing data for the reference forecast only, the appropriate exception was not being caught and thrown and the reading of other files (with valid data) did not continue. Instead, the whole I/O failed indicating that no data could be read for the reference forecast, due to no data being accepted from a single file. The code in PublishedInterfaceXMLIO.Java was updated to throw a ConditionException.
- Fixed a bug in the identification of VUs to include in AUs whereby the VerificationUnit.aggregationEquals method was testing the quality of the support for the forecasts and observations using the equals method (exact equality) rather than the evs.analysunits.scale.TemporalSupport.equalsOnUnitChange method (equality) after any requested changes in measurement units. This led to valid candidates being rejected for having different support, despite the equivalent support after the requested unit change was implemented.
- A memory leak was discovered in the EVS when using a large number of stations (e.g. > 100). This stemmed from a persistent reference to paired data within each VU, after the VU had been processed. The logic for retaining the pointer was to allow for quicker calculations when subsequently processed VUs depended on earlier units (e.g. for skill scores or aggregations across locations). The code was improved to release the pointer to old pairs when no longer needed.

6.4 Feature upgrades and modifications related to Graphical User's Interface

- Re-labeled the "Start" button in the GUI Output dialog to read "First." Previously, “Start” was confused to mean generate output (“Run”).
- Removed the individual "Save" buttons from each GUI window, as the "Save" option is always accessible from the task bar. This avoids confusion about when to save a project.

- When exiting the GUI, the options now include saving the current project, not saving the project, or cancelling. Previously, only exit (without save) and cancel options were available, potentially leading to the inadvertent failure to save a current project.
- Added warning messages on setting the date formats for ASCII whereby case sensitivity in the date formatting could be missed, inadvertently, leading to the incorrect interpretation of dates (e.g. m versus M for minute versus month). This required additional warnings in AnalysisUnit.java, ASCIIFileIO.java, OHDFileIO.java and a new method in StringUtilities.java to check the dates.
- Added several options to the MoreVerificationWindowDialog.java to allow for conditioning on dates that represent forecast issue times as well as valid times.
- Added parameter support and plotting options for a new verification metric, namely the rank histogram.

6.5 Feature upgrades and modifications not related to Graphical User's Interface

- Allowed for conditioning of verification pairs with date conditions that comprise forecast issue times as well as valid times.
- Added the rank histogram to the available verification metrics in the EVS. Tied ensemble members are treated by randomly sampling from the tied ranks. By default, the rank histogram comprises the relative frequencies (probabilities) that an observation falls between two ensemble members, in order to allow simpler comparisons between thresholds or forecast datasets with different sample counts. However, an advanced option allows for the sample counts to be expressed in absolute units rather than empirical probabilities. When the rank histogram is computed for a single dataset, without the use of thresholds to subset data (i.e. conditional rank histograms), a traditional histogram is used. When multiple subsets of the original sample data are used (i.e. thresholds), the resulting “conditional rank histograms” are plotted with lines for improved visibility.
- Allowed for simple transformations of the EVS XML outputs using an Extensible Stylesheet Language Transformation (XSLT). The transformation is implemented on the command line using a XSLT style sheet. This simplifies the use of the EVS outputs in secondary applications, which require a different format than the native format output by the EVS.

- Added a minimum sample size constraint when computing metrics. If the sample size falls below the minimum constraint, the metric is not computed. The precise meaning of that constraint varies between metrics. For example, when computing dichotomous metrics (i.e. metrics computed with respect to discrete events), such as the Brier Score or reliability diagram, the minimum sample size refers to the smaller of the number of occurrences and non-occurrences of the event. For continuous metrics, such as the mean error or the CRPS, it simply comprises the number of (possibly conditional) samples from which the metric was computed. This option is only available via the EVS project file using the XML tag <minimum_sample_size_parameter> for a chosen metric, together with an integer value for the minimum sample size.
- Allowed custom attribute units to be defined (previously, this was constrained to units in the EVS unit library).
- Expanded the default attribute units available in the EVS unit library.
- When climatological thresholds are specified that refer to duplicate real-valued thresholds, these duplicates are no longer computed.
- Updated PairedData.java to store verifying observations locally in order to avoid re-reading them from file. Also, separated between the verifying observations read from file (the unconditional observations) and those associated with the paired data. This allows for an unconditional climatology to be stored locally also. An immediate benefit was a reduction in the run times when conducting verification with large observed datasets.
- In order to circumvent the high memory usage by the EVS for those forecast locations where the forecasts are much more resolved (e.g. hourly) than the temporal resolution required for verification (e.g. daily), an option was included to store the paired data in their aggregate (e.g. daily) resolution rather than their native resolution. Further, the forecast I/O was updated to allow for "on-the-fly" aggregation of the forecasts (i.e. after reading each file), thereby avoiding the need to read and store all forecasts at their native resolution.
- Reading of NWS binary files for "Conditional Simulations" (CS) was previously restricted to forecast data, not observations, yet the NWS binary files may contain simulated streamflows that are useful for verification purposes. Of the NWS formats previously supported, only CARD was supported for observations

and NWS binary for forecasts. The NWS binary (CS) format is now supported for verifying observations (which may comprise simulated streamflows).

- Expanded the range of options for allowing conditioning of the sample climatology in the EVS as a baseline forecast (i.e. the use of a “conditional climatology” as the baseline).
- Added new constraints for restricting the reading of forecasts and observations when conducting pairing. Previously, limited constraints were provided on a file-reader-specific basis for restricting the range of dates and lead times processed. This was abstracted into a separate class, ConditionArray.java, for storing an array of conditions. Constrained I/O is now implemented consistently across all file readers, simplifying the coding of further constraints in future. Currently, constraints are allowed on the period of record and on forecast lead times.
- Improved the memory management during pairing of very large datasets by PairedData.java (e.g. required for long-range hindcasting). Specifically, updated the sortTraceByTrace method that’s sorts the pairs in trace order, using a new ForecastTime.java object to automatically sort the pairs in a TreeMap, rather than using layered TreeMaps. The resulting code is marginally slower for small datasets, but significantly improves the memory management for very large datasets.
- Added several new temporal aggregation functions to the FunctionLibrary.java class.
- Abstracted an option to conduct strict temporal aggregation (ignoring data blocks with any missing data versus processing the non-missing data) from a class variable in PairedData.java to a method variable in getTimeAggData, allowing strict aggregation to be controlled via the appropriate method call, rather than being hard-coded to always perform strict aggregation.
- Added functionality in PairedData.java to handle NaN values, as well as missing values, and also updated the missing value identifier in VerificationUnit.java to ensure that NaN is not a valid missing value identifier. In the process, abstracted the search for missing and NaN values to more generic functions in FunctionLibrary.java, an example of which is the new VectorDoubleProcedure.java

- Added an option to specify the graphics output format on the command line by adding a graphics format tag underneath the <output_data> tag in the project file, namely <output_graphics_format>, with supported options comprising JPEG, PNG and SVG (Scalable Vector Graphics) and the default, JPEG.

6.6 Feature upgrades and modifications for developers

- Moved all non-source files from the /src directories to /nonsrc. This required some code improvements to locate parameter files that were previously located underneath the source tree. The refactored structure is cleaner, ensuring only source code is located underneath the source tree.

7.0 Changes from EVS 3.0 to EVS 4.0

7.1 Changes in default behavior

- The observed and forecast file types must now be explicitly defined. This avoids ambiguities and performance issues that can arise from auto-detecting the file types (by opening and reading in the first few bytes in the file and checking against expected types). The default data types are ASCII. Thus, old project files for which paired data are no longer available will encounter an I/O error message if the file types are not ASCII (requiring the types to be explicitly defined for the first time).
- Changed the metric type of the ROC Score from a test of the "Ensemble distribution" to a test of "Ensemble skill." Arbitrary reference forecasts are now supported for the ROC Score, not just sample climatology.

7.2 Bug fixes related to the Graphical User's Interface

- Corrected several broken hyperlinks in the html descriptions of the verification metrics, which are displayed in the second window of the "Verification" stage.
- Fixed a bug whereby the internal hyperlinks between the descriptions of the EVS verification metrics (i.e. from one html description to another) were not accessible. This was due to the packaging of these html descriptions within a single executable JAR file. In order for the internal links to work properly, a separate directory containing the html descriptions (outside of the executable JAR file) is required, and is now provided with the distribution, namely /evs/resources/statsexplained. The consequences of moving the executable JAR or not downloading this separate directory are minor (i.e. the internal hyperlinks will not work, and navigation to the appropriate description must be performed manually). However, an error message is now provided in that case, indicating that the html resource cannot be found locally.

7.3 Bug fixes not related to Graphical User's Interface

- Corrected the importing of NWS Card files to account for several forecast issue times within one day. Previously, the issue date was represented by the year, month, and day of month, but not the hour of day. This led to the duplication (and subsequent elimination) of verification pairs that were assumed, incorrectly, to originate from the same hour of the day.

- Fixed several bugs in the reading of PI-XML files, including the failure to read observations in PI-XML format.
- Fixed several minor bugs identified by FindBugs: http://findbugs.sourceforge.net/

7.4 Feature upgrades and modifications related to Graphical User's Interface

- Added the facility to specify the observed and forecast file types in the first verification window, via drop-down menus.
- Added the option to manually specify the date formats for ASCII observed and forecast files. These options appear in the "Other options" pane of the "Additional options" dialog. The "Additional options" dialog is accessed using the "More" button associated with the input data (2b) in the first window of the "Verification" stage.
- Added an option to specify the number of decimal places for writing verification pairs (the default is 5). This option appears in the "Other options" pane of the "Additional options" dialog. The "Additional options" dialog is accessed using the "More" button associated with the input data (2b) in the first window of the "Verification" stage.
- Added an option to use all observations when computing the sample climatology corresponding to real-valued thresholds. This option appears in the "Other options" pane of the "Additional options" dialog. The "Additional options" dialog is accessed using the "More" button associated with the input data (2b) in the first window of the "Verification" stage.
- Added an option to suppress the writing of conditional pairs via the GUI. Previously, this option was available in the project file only. It is now accessed using the "More" button associated with the output data (2d) in the first window of the "Verification" stage.
- Added the RME and associated explanation and parameter options to the second window of the "Verification" stage.
- Added an option to apply the thresholds defined for a selected verification metric to all other metrics associated with the current Verification Unit (VU). The IDs and status of the thresholds as "Main" (i.e. thresholds for display) are also copied. When copying thresholds that comprise an unconditional constraint (i.e. use of "All data"), the unconditional threshold is only copied to metrics that are based on discrete events. The feature is accessed via the "Do all" button under the basic parameter options (3c) in the second window of the

"Verification" stage (after selecting a metric that comprises thresholds as a basic parameter option).

- Reorganized and improved the display of additional parameter options for the verification metrics. These are accessed via the "More" button from the basic parameter options (3c) in the second window of the "Verification" stage.
- Added options relating to the calculation of confidence intervals for the verification metrics. These include specifying one or more intervals in the range [0,1], together with the parameters of the stationary block bootstrap algorithm. These options are included in a separate tabbed pane (labeled "Confidence intervals") under the advanced parameter options for a particular metric. The advanced parameter options are accessed via the "More" button from the basic parameter options (3c) in the second window of the "Verification" stage. Additional options apply to the calculation of confidence intervals for AUs. These options are accessible via the "More" button, which is located adjacent to the tabulated list of VUs (2b) in the "Aggregation" stage. The options include omitting the calculation of confidence intervals for the AU (regardless of whether they are required for particular VUs) and specifying whether the component VUs are statistically dependent.
- Added the option to pool verification pairs across several forecast locations and to compute the verification metrics from the pooled pairs, rather than averaging the metrics from the individual locations. Previously, this option was only accessible via the XML project file. It is now accessible via the "More" button, which is located adjacent to the tabulated list of VUs (2b) in the "Aggregation" stage.
- Added shortcuts for selecting particular combinations of metrics in the products table (1b) of the "Output" stage. These options are accessed by right-clicking on the products table. New options include the ability to select all forecast lead times for the selected metrics across all VUs (e.g. to output results for the correlation coefficient across several VUs).
- Improved the error handling and reporting code, providing more specific and detailed error messages.

7.5 Feature upgrades and modifications not related to Graphical User's Interface

- Added the facility to manually specify the date formats used in the ASCII observed and forecast files. A date format is based on the elementary components, yyyy (year), MM (calendar month), dd (day of month), HH (hour of

day in the 24-hour clock), mm (minute of hour) and ss (second of minute). The elements are separated with single characters or whitespace (e.g. MM/dd/yyyy HH) or appended without separators (e.g. yyyyMMddHH). The default date format is MM/dd/yyyy HH.

- Added the relative mean error (RME) to the set of deterministic metrics available in the EVS. The RME comprises the mean error as a fraction of the mean observed value over the sample.
- Added an additional method for computing the Area Under the Curve (AUC) for the empirical Relative Operating Characteristic (ROC) Score. The default method remains the algorithm described in Mason and Graham (2002): Mason, S.J. and Graham N.E., 2002: Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation, *Quarterly Journal of the Royal Meteorological Society*, **30**, 291-303. The alternative involves computing the AUC from the empirical ROC curve, which is based on a finite number of points/classifiers. The integral of the empirical ROC curve (AUC) is computed using the trapezoid rule. In most cases, the algorithm described by Mason and Graham (2002) generates larger values of the AUC (skill) than the integral of the empirical ROC curve.
- Extended the ROC Score to allow for arbitrary reference forecasts, not just sample climatology.
- Added the likelihood-base-rate (LBR) decomposition of the Brier Score into Type-II conditional bias, discrimination and sharpness, and the corresponding decomposition of the Brier Skill Score into relative Type-II conditional bias, relative discrimination and relative sharpness.
- Added options for computing confidence intervals for the verification metrics (except the box plots). One or more confidence intervals may be computed for selected metrics of a VU and for an AU. The confidence intervals are derived from a stationary block bootstrap of the verification pairs. In order to account for temporal statistical dependence, the bootstrap resampling applies to contiguous “blocks” of pairs rather than individual pairs. The blocks are parameterized by their mean length and are sampled from a geometric distribution with that parameter. In order to account for spatial dependence (when computing aggregate verification results across several locations), the absolute times of the sampled blocks may be coordinated/fixed across all

locations. The bootstrap algorithm is multi-threaded for improved performance on multi-core, multi-processor, machines.

- Provided additional command line options for suppressing the writing of either the graphical or numerical outputs when running the EVS in batch mode. The writing of graphical outputs is suppressed with –g and the writing of numerical outputs is suppressed with –n.
- Provided additional command line options for converting between legacy file formats (NWS Card and NWS CS binary) and a generic ASCII file format used by the EVS. The option –bin2asc in.cs out.fcst converts the NWS CS binary file, in.cs, to the ASCII output file, out.fst. The option –fcard2asc in.fcst out.fcst converts the forecast data in NWS Card file, in.fcst, to the ASCII output file, out.fst. The option –ocard2asc in.obs out.obs converts the observed data in NWS Card file, in.obs, to the ASCII output file, out.obs.
- Improved the R utilities script for reading the XML numerical outputs from the EVS into R (www.r-project.org). The utilities script is located in /evs/resources/rscripts/Utilities.R. There are three key methods for reading the different EVS outputs, namely readEVSScores, which reads the verification scores, readEVSDiagrams, which reads the verification diagrams (e.g. reliability diagram, spread-bias diagram) and readEVSBoxPlots, which reads the EVS box plots.
- Provided two self-contained R scripts in /evs/resources/rscripts/example_scripts to demonstrate the plotting of EVS output in R. The necessary EVS (XML) outputs are located in evs/resources/rscripts/example_scripts/example_evs_out.
- Added an option to omit 'no-data' values from the paired files.

7.6 Feature upgrades and modifications for developers

- Abstracted control of metric calculation by lead time from the individual metrics to a new method in the evs.metric.metrics package, namely computeByLeadTime. This leads to much cleaner and more extensible code. For example, it is no longer necessary to iterate through lead-times when implementing the compute method of a new metric. Rather, the compute method is now generic for the given input data. There are several downstream effects of these changes that have led to significantly better performance, particularly when computing bootstrap confidence intervals (i.e. repeatedly calculating the metrics).

8.0 Changes from EVS 2.0 to EVS 3.0

8.1 Changes in default behavior

- Changed the temporal aggregation default to store the maximum valid time of the input times (in UTC) when conducting aggregation. Previously, the default was to compute the mean of the input times. The default for handling the forecast lead times remains to compute the maximum of the inputs. Thus, for example, aggregation of four six-hourly pairs at increasing UTC times of {18, 0, 6, 12} previously generated an aggregated paired value with time UTC 3, but will now generate an aggregated paired value with time UTC 12 (note that 12 proceeds 18 when considering date). Thus, the aggregated value should be interpreted as the value over the period of aggregation immediately preceding the stated time.
- Changed the start and end dates of the verification period defined in the first verification window from the forecast time zone to UTC. The start and end dates begin and end at 00 UTC on the specified date, respectively. Thus, in order to include verification pairs that fall on the specified end day, one day should be added to the input date.
- Changed the order of error messages displayed in the GUI Console window (not to be confused with an external console) so that the latest error messages are displayed at the top of the console rather than appended to the bottom.

8.2 Bug fixes related to Graphical User's Interface

- Fixed a bug in the table of thresholds for each verification metric, ensuring that the scrolling window expands properly as new thresholds are added (previously a fixed limit).
- Fixed a bug in the table of candidate units for aggregation, ensuring that the scrolling window expands properly to show all available units (previously a fixed limit).
- Corrected a bug in the GUI for selecting pre-conditions to apply to the verification pairs. Entering incorrect conditions lead to a (correct) warning, but, when subsequently cancelling further edits, the existing (valid) conditions were removed rather than being returned to the original (valid) state.

8.3 Bug fixes not related to Graphical User's Interface

- Fixed a bug in the BSS and CRPSS to write the null value identifier where the output of the BSS or CRPSS is undefined (i.e. divide by zero), thereby allowing proper display in the plots for those lead times where the score *is* defined.
- Fixed a bug in the reading of an XML paired file where one or more ensemble members were missing, i.e. the number of members in the paired file (which does not store missing members) varied. The result of this bug was that missing members were initialized with the java default value for a float, namely 0.0, and not the pre-defined null value identifier.
- Fixed a bug in the skill score calculations (BSS and CRPSS). When these calculations were performed repeatedly (i.e. by clicking “Run” two or more times), the skill scores were zeroed on the second or further runs. While the metrics would not normally be re-computed, the bug has been fixed.
- Fixed a bug in the application of value conditions, whereby the conditions were being applied *before* any requested changes in measurement units rather than *after*. The value conditions (and hence real-valued thresholds on which those conditions are based) are now in the target measurement units.
- Fixed a bug in a method for ordering the paired data by trace, whereby the last trace (in order of forecast valid time) was not being appended to the results and, therefore, included in the verification results.
- Spell-checked all developer documentation and corrected spelling mistakes.

8.4 Feature upgrades and modifications not related to Graphical User’s Interface

- Packaged the EVS into a single executable JAR file with all associated libraries using an ANT build script. The EVS is now delivered as a single executable JAR file, without the need to maintain an internal directory structure for dependent libraries (which are now packaged and accessed from within the executable JAR).
- The writing of conditional pairs (i.e. a subset of the overall pairs with any conditions on variable value or date applied) has been made optional to speed-up the processing of large numbers of verification points. This option is implemented via the <write_conditional_pairs> tag of the <paired_data> section of the EVS project file, with a default of true, i.e. conditional pairs are written by default, as before. This option is also accessible via the GUI (see below).

- Improved the performance of the temporal aggregation routine and provided options for the type of aggregation function applied not only to the forecast data, but to the forecast valid times and lead times. Also changed the default behavior; previously, the forecast valid time was given the mean of the input times and the lead time assumed the maximum of the input times; the forecast valid time now assumes the maximum of the input times (see below).
- Improved information and error messages printed to standard out (i.e. the console, if EVS was initiated from a console window).
- Added options for aggregating the support of the observed data to match the support of the forecasts, including the ability to compute an accumulation over a forecast window. Eventually, an aggregation function will be implemented for every possible combination of input support allowed in the EVS (including a change of measurement units).
- Added methods for reading paired data in the same ASCII format to which the XML pairs can be converted. Thus, paired files may be produced and read in ASCII format (as well as forecasts and observations) where convenient.
- Implemented additional R scripts for plotting the verification results produced by the EVS (i.e. the XML output), including a script that will plot the EVS single-valued metrics and ensemble scores as a function of threshold value.
- Implemented the three-part decomposition of the Brier Score into: Brier Score = reliability – resolution + uncertainty and the associated graphical and numerical products.
- Implemented additional options for averaging the forecast ensemble members when computing single-valued verification metrics such as the mean error, root mean square error, mean absolute error, and correlation coefficient. The default remains to compute the ensemble mean. Additional options now include the ensemble median and mode.
- Added the Mean Absolute Error of the ensemble average to the single-valued verification metrics.
- Added the climatological frequency and the zero-skill line to the reliability diagram (located half-way between the climatological frequency and the expected frequency for a reliable forecasting system). These curves are available in the XML output files, but are not plotted within the EVS.
- Backwards compatibility has been maintained for earlier project files. Upon saving old project files within a new version of EVS, new options will be written with their default values.

- Implemented the concept of “main” and “auxiliary” thresholds for metrics that either require or support thresholds. Currently, the “main” thresholds are used to identify events (or subsets of data) that should be included in the graphical outputs from the EVS. By default, both the “main” and “auxiliary” thresholds are included in the numerical outputs from the EVS. This information is stored as an additional XML tag in the project file, labeled main_threshold, which contains a list of Boolean values equal in length to the number of thresholds (true indicates a main threshold).
- Added the facility to derive climatological probability thresholds from a larger set of observed data. In the EVS Version 2.0 they were derived from the paired observations, after applying any requested changes in units, temporal aggregation, or value and date conditions, including the discrete verification time-period requested in the first verification window. Now, they may be derived from the original observed data, again after applying any requested changes in units, temporal aggregation, or date and (observed) value conditions, but NOT the discrete verification time-period (i.e. the full period of record covered by the observed file will be used, after applying any changes in measurement units, temporal aggregation, value conditions on the observations, and date conditions *except* for the reduced verification time period). This option is controlled by a Boolean flag, which is accessible via a check box in the GUI, and also in the project file under the new XML tag, labeled use_all_observations_for_climatology. Note that this option only applies to the derivation of real-valued thresholds corresponding to particular climatological probabilities of occurrence. For those metrics that incorporate climatological probabilities in the calculation (e.g. the climatological frequency in the reliability diagram), the behavior is unchanged (the observations associated with the conditional pairs are still used).
- Added further date and value (pre-)conditions, including additional statistics for selecting forecasts based on value (ensemble median and mode, probability of not exceeding a given value, and the value corresponding to a given non-exceedence probability) and additional functions for selecting forecasts and observations based on dates (hours of day in UTC). Thus, much more complex pre-conditions are now possible, such as selecting only those pairs (for verification) whose forecast probability of exceeding “flood stage” is greater than 0.9. This functionality is necessary for real-time verification, where the aim is to select historical (observed and forecast) analogs to a real-time

forecast based on specific properties of that real-time forecast, and possibly auxiliary information.

- Added the facility to compute the binormal approximation to the Relative Operating Characteristic (ROC) curve and the associated ROC Score. The approximation is based on fitting the binormal model to the empirical (POD, POFD) pairs and is, therefore, dependent on the number of thresholds chosen. For an exact comparison between the binormal approximation to the ROC curve and the binormal approximation to the ROC Score, a common number of thresholds should be adopted for each metric. However, when comparing the empirical ROC Score to the binormal ROC Score, the results will be closest when adopting *m*+1 thresholds, where *m* is the number of ensemble members per forecast. Specifically, the empirical ROC Score is derived from ranking of the POD and POFD data, rather than computing the ROC curve. The ranked data can take at most *m*, "jumps" in probability (at the corresponding ensemble member positions). Hence, the empirical ROC Score is analogous to deriving the ROC Score from an empirical ROC curve constructed with *m*+1 thresholds.
- Added the facility to aggregate the observed support prior to verification. Previously, this was only possible for the forecasts. The same restrictions apply to change of support of the observations as the forecasts, namely the aggregated support is exactly divisible by the frequency of the data and comprises either a mean of the input values if the inputs have instantaneous support or a total of the input values if the input values are totals.
- Added the option to remove certain lead times from the verification results based on a minimum sample-size requirement. The sample size constraint is set by a fraction in the range [0,1]. The fraction is multiplied by the average number of pairs across all lead times to determine the minimum sample size in numbers of pairs. For example, a fraction of 0.5 implies that verification results will *not* be computed for any lead time with fewer than 50% of the average number of pairs across all lead times.

The following new features are only accessible via the EVS project file (not the GUI):

- Added the facility to specify the method for computing CRPS in the EVS project file. By default the Hersbach (2000) method is used, but a method that can handle null ensemble members has been added. This is specified in the <crps_method> tag of the crps metric in the EVS project file, with options

hersbach and with_nulls. If null members are present, the hersbach option will lead to all forecasts with one or more null members being removed from the calculation, and will fail to compute if all forecasts contain one or more null members.

- The facility to prevent elimination of duplicate pairs (pairs with a common forecast valid time and lead time) has been added to the EVS project file. This is implemented via the <eliminate_duplicates> tag of the <paired_data> section of the EVS project file, with a default value of true. When false, duplicate pairs will not be eliminated. This is necessary when computing verification metrics for data that have been pre-pooled across several forecast locations and contained in a *single* paired file.
- Added an option to pool the verification pairs across several forecast locations and to compute the verification metrics from the pooled pairs, rather than averaging the metrics from the individual locations. Theoretically, this approach is preferred, but is much more time-consuming in practice, and is usually not feasible. The default behavior remains to average the verification metrics from the individual locations. The new option is only accessible via the EVS project file by adding or setting the <pool_pairs>true</pool_pairs> entry to the XML for a particular AU, where true implies that pooling will be conducted, and false implies averaging.
- Added the facility to change the decimal writing precision of the paired data for a given VU. This applies to both the raw and conditional pairs. The default behavior is to write forecasts and observations with a maximum precision of five decimal places (unchanged), with fewer places written as required. The integer number of decimal places (>0) can now be defined in the project file using the <paired_write_precision>5</paired_write_precision> tag, which is part of the <paired_data> block. This functionality is not accessible via the GUI, and existing pairs will not be re-written with a new decimal precision (unless re-writing is otherwise necessary).
- Added the facility to set the behavior for removing null ensemble members when writing the paired file. The default behavior remains to *remove* null ensemble members. This may be changed using the <strip_nulls_from_paired_file>true</strip_nulls_from_paired_file> tag, which is part of the <paired_data> block. This functionality is not accessible via the GUI, and existing pairs will not be re-written (unless otherwise required).

8.5 Feature upgrades and modifications related to Graphical User's Interface

- Simplified the behavior of the reference forecast selection for skill scores. If a skill score is selected and only one possible reference forecast is available, this is automatically selected in the secondary parameters dialog.
- Added a menu to the VU table in the Output window to allow for the (de)selection of all products and lead times for all available units.
- Added advanced options for computing different averages from the ensemble members when using single-valued verification metrics.
- Added auto-recall of the last directory accessed when creating, saving and reading project files so that the last working directory is opened by default.
- Added an option to change the behavior for writing conditional pairs. This is controlled by a check button in the advanced options (accessed via the “More” button) in the “Output” section of the first window in the GUI. The default (slower) behavior is to write conditional pairs.
- Added support for multiple-row selection and deletion in the thresholds table associated with each verification metric (in the second verification window).
- Added the facility to distinguish between “main” and “auxiliary” thresholds for metrics that either require or support thresholds. By default, the “main” thresholds are used for plotting and all thresholds (“main” and “auxiliary”) are added to any numerical outputs written by the EVS.
- Added functionality to generate verification thresholds semi-automatically for verification metrics that either require or support thresholds. The thresholds are generated by entering a number of thresholds, the first threshold value, and a constant increment (positive or negative) between thresholds. This is useful for designing plots that show verification scores as a ‘continuous’ function of threshold value (i.e. outside of the EVS). By default, only those thresholds identified as “main” thresholds are included in the graphical outputs from the EVS, but the numerical outputs (on which custom plots are designed) include all of the thresholds.
- Re-labeled the “Edit no-data value” option in the advanced input data options dialog to the more generic “Edit other options” and added a check box to control the way observed data are used to determine climatological probability thresholds.
- Improved the GUI for selecting pre-conditions to apply to the verification pairs, in keeping with the enhanced functionality for identifying pre-conditions (see above).

- Added option to iconify (or “minimize” in Windows terminology) the EVS GUI while processing a verification project; a button labeled “iconify” has been added to the progress dialog to facilitate this.
- Changed the label Forecast lead period to Forecast lead time horizon in the first window of the GUI.
- Added the binormal approximation to the ROC and ROC Score to the GUI. In both cases, the binormal approximation is appended to the results when selecting to do so under the Advanced Parameter Options dialog. In that case, the empirical ROC data are plotted as open points and the binormal approximation is plotted with a line of the same color.
- Removed the text (in forecast time system) for both the start and end dates of the verification period in the first window of the GUI, reflecting the change to UTC (see above).
- Added an error message when attempting to temporally aggregate forecasts over a longer period than the specified forecast lead time horizon (e.g. attempting to compute monthly averages for forecasts with a lead time horizon of 14 days).
- Improved the error console in the GUI.

8.6 Feature upgrades and modifications for developers

- Added numerous additional methods for developers that assist in sub-setting paired data according to varied conditions met in particular rows or columns of the paired-data matrix. These methods can be found in evs.utilities.matrix.DoubleMatrix2D. The conditions can be made arbitrarily complex by chaining together functions provided in the evs.utilities.mathutil package and applying them to specified rows and/or columns in the paired data matrix.
- Added a method to linearly interpolate observed data to the nearest forecast valid time using a weighed (by temporal separation) combination of the two nearest observations between which the forecast valid time lies. The method is: evs.data.PairedData.linInterpObsToFcsts. This functionality is currently only accessible to developers, but may be included in the GUI in future.
- Enhanced the processing of timing information by adding a dedicated class for representing forecast valid times and lead times: evs.data.ValidTime.

9.0 Changes from EVS 1.0 to EVS 2.0

9.1 Feature upgrades and modifications related to Graphical User's Interface

- Removed Time-Series ID and renamed River Segment ID to Location ID.
- Moved basic output options from pop-up window to main Output window.
- Implemented enhanced error dialog with improved error messages.
- Implemented enhanced progress monitor to monitor and display progress of paired-file reading (and included a pair count in the paired file to enable this).
- Updated the aggregation window to include a weighting input in the table of verification metrics. Also renamed some features in this window.
- Added the option to select an arbitrary reference forecast for a skill metric.
- Added the option to show skill score decompositions in a tabbed pane (similar to metrics with one plot per lead time), which may be animated.
- Improved display of zero error line in plots (extended continuously).
- Improved auto-scaling of axes in plots.

9.2 Feature upgrades and modifications not related to Graphical User's Interface

- Added multiplication factor in support dialog to allow simple changes between measurement units (more complex operations, such as a change in temperature units, are not yet supported).
- Implemented reading of PI-XML observations
- Implemented reading of PI-XML forecasts
- Implemented reading of ASCII observations
- Implemented reading of ASCII forecasts
- Changed representation of forecast lead times from integer hours to double-precision float hours to allow verification of forecasts with lead times in fractional hours, thereby extending the EVS to arbitrary forecast lead times.
- Rewrote the online documentation and updated the mathematical formulas for all of the verification metrics.
- Implemented an R script for each metric in the EVS to read in the XML output and produce high quality plots in EPS format for scientific papers.
- Modified calculation of the mean CRPS to account for the relative position of the observation between the two adjacent ensemble members.

- Added ROC score to the available metrics and included a plot by forecast lead time (same as with other scores). The calculation is based on Mason and Graham (2000).
- Added a sample size metric and associated plot to compute the basic sample size information by forecast lead time and threshold. This may be used for exploratory purposes before computing other verification metrics. In future, we may add further metrics for data exploration (of the observed and forecast data rather than the verification pairs).
- Added modified box plot by size of observed value to GUI (previously via the command line only).
- Modified the spatial aggregation routine to compute the expected (mean) value of each metric across a set of VUs rather than pooling paired data.
- Included ability to perform a weighted spatial aggregation. The weight is uniform by default and must sum to 1. A non-uniform weight is also permissible and a weight of "S" is used to weight by the sample size at the first lead time (i.e. maintaining constant weights across lead times). If a verification metric is not available for a given lead time or threshold the weights are automatically rescaled to sum to 1, maintaining the correct relative weighting of the available metrics.
- Relaxed constraints on spatial aggregation to allow aggregation for VUs with different start and end dates.
- Improved the efficiency of file reading for external file formats to ensure that only data within the specified start and end dates and forecast lead times are fully read (otherwise only the file headers are read to check this information).
- Implemented backwards compatibility for the above features so that they do not prevent running of old EVS projects. However, the aggregation routine and CRPS update has been swapped without the option of backwards compatibility. Thus, old projects with spatial aggregation will produce different results in the EVS 2.0. The CRPS update was a bug fix, voiding the need for backwards compatibility.
- Updated the algorithm for CRPS to the method described in Hersbach, H., 2000: Decomposition of the Continuous Ranked Probability Score for Ensemble Prediction Systems. There are small numerical differences between the old and new algorithms. Also, the procedure described in Hersbach assumes that a constant number of ensemble members is available, whereas the previous method for computing CRPS had no such constraint. Thus,

differences will be seen when comparing numbers between systems for which some forecasts comprise null ensemble members.

- Added the decomposition of the CRPS into reliability, resolution and uncertainty, as described in Hersbach (2000).
- Added the Brier Skill Score (BSS) for an arbitrary reference forecast selected by the user.
- Added the Continuous Ranked Probability Skill Score (CRPSS) for an arbitrary reference forecast selected by the user.

10.0 Changes from EVS 1.0 BETA to EVS 1.0

10.1 Feature upgrades and modifications related to Graphical User's Interface

- Removed the table containing reference forecasts, which are not yet enabled.
- Improved the labeling of various options (e.g. 'time zones' rather than 'time systems').

10.2 Feature upgrades and modifications not related to Graphical User's Interface)

- Allowed real-valued thresholds for all metrics.
- Allowed probability thresholds for all metrics, not just Brier, ROC, and Reliability.
- Included the option for thresholding with a closed interval (i.e. a "between" condition).
- Supported the use of symmetric windows around the forecast median in the Spread-Bias plot.
- Included sharpness (sample-count) plot in the Reliability diagram.
- Changed the definition of probability thresholds in Spread-Bias plot, Mean Capture Rate diagram and Box Plots. Previously, these thresholds referred to plotting positions (i.e. plot resolution) and NOT thresholds of the observed distribution. They now refer to thresholds of the observed distribution for consistency with all other metrics. Plotting positions are now determined with a 'points count' parameter. For example, a point count of 10 for the Spread-Bias plot will construct a plot comprising 10 points.
- Added a new 'points count' parameter for the Spread-Bias plot, Mean Capture Rate diagram, Box Plots and Reliability diagram, which allows the resolution of those diagrams to be altered.

- Included the option to change the default temporal aggregation function from the mean over a specified period to one of several other functions, including the minimum, maximum, and total (i.e. accumulation).
- Included units in the plots that comprise real units (mean error, RMSE, Mean Capture Rate diagram, box plots) once those units have been added to the observed and forecast support for a VU.
- Included an option to animate a sequence of verification graphics at different lead times.
- Included writing of sample counts to an XML file when writing other numerical results.
- Included writing of conditional pairs to XML format as well as the original pairs.
- Included an option to ignore global value conditions on a per-metric basis. For example, if a condition was applied to consider only those pairs whose ensemble mean temperatures exceeded freezing, this condition could be ignored on a per-metric basis.
- Enabled backwards compatibility with old project files (i.e. projects with old options will run as before).
- Enhanced and updated documentation.
- Improved memory management for AUs.